Experiences with digital pen, keyboard and mouse usability

Beryl Plimmer
Department of Computer Science
University of Auckland
Auckland, New Zealand
+64 9 373 7599
beryl@cs.auckland.ac.nz

ABSTRACT
Digital pens provide nice, natural human computer interaction for tasks such as annotating documents and sketching. However interfaces that use a pen alone can be slow and inefficient. Thus most pen interfaces also support keyboard and mouse input. Multi-modal input exponentially increases the complexity of the design and usability of these systems. Here we describe our usability testing experiences of four different pen-dominant software tools. One is designed for a digital whiteboard, two for a Tablet PC and the last for a Tablet PC coupled to a haptic pen. Our experiences may be of interest to others working with pen-based software and multi-modal interfaces.

Categories and Subject Descriptors
H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces. - Graphical user interfaces

General Terms
Design, Human Factors.

Keywords
Usability testing, multi-modal interface, pen-based computing.

1. INTRODUCTION
Pen and inking is a natural and appealing way to record ideas and pen based interfaces are increasing in popularity as better hardware becomes available. However our experiences with designing and building these systems on standard operating system components suggest that there are many usability issues that differ from the standard interaction devices.

There are arguments for designing pen-based interfaces to work solely with pen input [4]. A pen only interaction space improves mobility and convenience as the user has only one input device to consider. However, in our experience, there are many practical obstacles to providing efficient, easy to use pen-only input. In essence these problems are a result of replacing a keyboard and mouse with one pen. A keyboard has approximately 70 discrete keyboard keys that can be combined with control keys to provide several hundred unique input codes and a mouse is a pointer with one or more supplementary buttons.

In contrast a pen is both a pointing device and input device. Pen input is often in the form of gestures that need to be recognized and recognition results are often inaccurate. The pen becomes an overloaded, unreliable device. Some of the challenges we have encountered include: accurate text input, unsuitable standard controls and dialogue boxes, pen button usability.

Text input via a pen can be achieved in two ways: on screen keyboards, which are slow and error prone; or handwriting supported by a recognition engine, which is quicker, but also error prone. With careful design these problems can be mitigated, however most standard dialogue boxes such as file open/save rely on text entry. Many other standard interaction techniques on the Microsoft Operating systems (the operating systems with most pen support) are accessed via right mouse buttons. The buttons on pens are difficult to use. As a result of these limitations many pen-dominant software tools also support keyboard and mouse input.

Much of the software for multi-modal systems, including pen-based systems, is experimental. In addition many of these systems are designed for mobile use. Both of these factors add to the confusion – the designers, users and usability testers do not have preexisting knowledge of how thing should be done. Usability testing mobile devices poses problems around environment and context of use.

We have developed and usability tested a number of different pen-based systems. Here we will present our usability testing experiences of four of these systems: Freeform a digital whiteboard user interface design tool; InkKit, a Tablet PC sketch toolkit; Penmarked a Tablet PC document annotation system and, lastly, a haptic pen writing training system for visually impaired users. We will briefly describe the purpose and technology for each system and then its usability testing. Following this is a discussion of the general challenges of multi-modal usability testing that we have encountered.

2. FREEFORM
Freeform is a pen-based design environment for user interface design. It is integrated into the Visual Basic 6 IDE (VB) and runs on standard Microsoft Windows operating systems [14-16, 18]. It was designed specifically as a collaborative design environment using digital whiteboards as the main interaction space (Figure 1). This large interactive display (Lids) is created by projecting the screen output onto an opaque glass screen that has a Mimio [6] pen capture bar attached to the side.

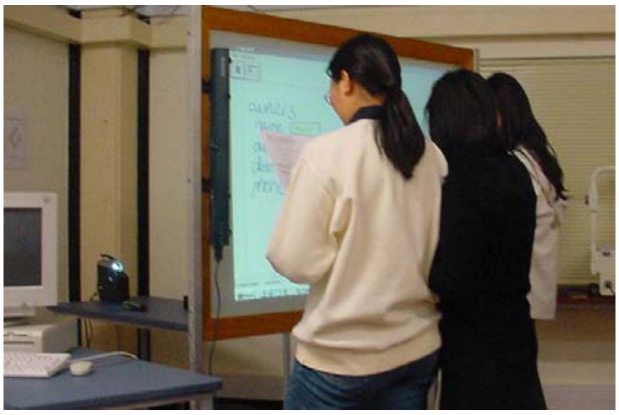


Figure 1: Freeform User Environment

The software has two main interfaces; a sketch canvas for sketching VB forms (Figure 2) and a storyboard for arranging the sketches (Figure 3). Because of the limitations of the recognition engine the sketch canvas has two inking modes – writing and drawing, there are also modes for editing and executing. The goal of the Freeform project was to provide a low-fidelity sketch interface for designing user interfaces and then compare Freeform with traditional whiteboard design and the VB form designer.

The software was designed so that as much as possible, the interaction was on the whiteboard using the Mimio pen as the interaction device. There is a facility on the Mimio capture bar to simulate a right mouse button, however we found this very difficult to use so did not implement any functionality dependent on it. The keyboard is used to enter text into standard dialogue boxes for file load/save as, at the time Freeform was developed (2001) there was no pen support in the operating system. As a part of the development process we completed two iterations of design, implementation and usability testing.

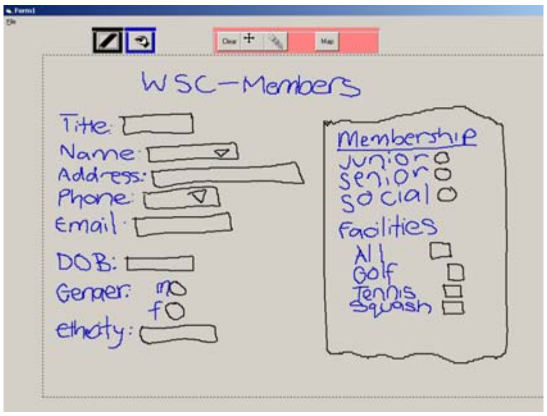


Figure 2: Freeform Form Sketch

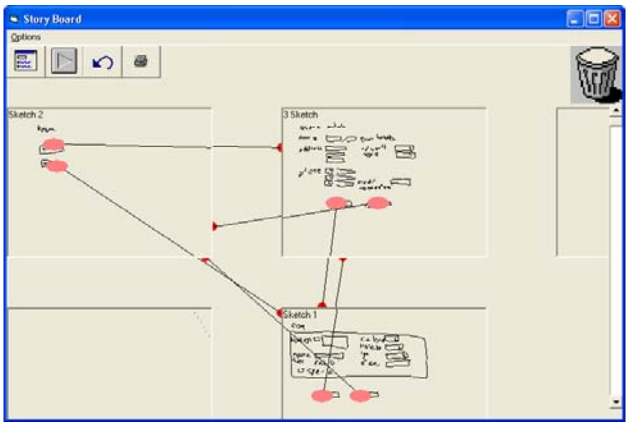


Figure 3: Freeform Storyboard

The usability studies looked at the central component of the system: the main sketch-space and resulting VB form. Nielson's [7] 'discount usability study' methodology was adopted. Students were asked to design a form for membership details for a sports club (Figure 2). They were required to sketch their design on the Lids screen, check its interaction by 'executing' the sketch and then use the recognition engine to create of a VB form. The main questions for the usability study were: Do people find the hardware usable? How easy is it to use the sketch-space and is the resulting VB form likely to be useful? What other features are required?

The sessions were recorded with two video cameras and screen captures, all pen actions were recorded and we observed the sessions. After the sessions we reviewed the observation notes and analysed the tapes to identify interaction problems. All users created over-sized diagrams; they told us that this was because of the size of the Mimio pen. Freeform supports resizing in edit mode by selecting a group of strokes. The selected strokes are surrounded by a box with corner grab handles. Because of the parallax error with the glass it was very difficult for users to grab the handles. We doubled the size of the handles between the 3rd and 4th usability tests.

There were also problems with the multi-modal nature of the interface with users writing in drawing mode and drawing in writing mode: as the separate writing and drawing modes were required for the recognition engine we added functionality to select ink and change its mode. The software did not require any right click or double click actions, however the operating system did. Right click required a button on the Mimio to be held down, this made it difficult to use. Double click requires both the clicks to be on exactly the same pixel – doing this successfully with a Mimio pen requires quite a bit of practice. User tended to resort to the keyboard and mouse to complete double-click or right-click tasks rather than persist with the pen.

The recognition engine is an implementation of Rubine's [20] algorithm used in the drawing mode to recognize shapes. In writing mode primitive word recognition is supported by combining Rubine's algorithm with a vocabulary list. The recognition rates for shapes are about 80%. Word recognition is much lower; however it is easy to replace an incorrect word by picking from a list. The expectations for word recognition were much lower when this study was conducted (2002) and the participants expressed satisfaction with the success rates. Further information on this project can be found else where [14, 15, 17, 18].

3. INKKIT

InkKit is a general sketching environment with a recognition engine for recognizing diagrams such as user interface designs, graphs and hierarchy charts [2, 3, 11]. There are two views of the designs, a portfolio that shows all the current sketches and individual sketches. InkKit has gone through several iterations; here we will highlight the major usability testing initiatives and outcomes.

InkKit was designed to leverage the pen support on Tablet PCs using the Tablet OS. Tablet PCs provide a higher fidelity interface for collecting ink data. These notebook computers have pen sensitive screens. Initially (2002) they ran a special version of the Microsoft Windows XP operating system. It included character recognition and a supplementary on screen keyboard/text input panel. We have observed some changes in the pen support in the OS over the last few years. For example, Tablet pens have a barrel button for right-click; however this has proved hard to use so has been supplemented with a push-and-hold-down action which triggers a right-click option. In addition, when standard text entry boxes get focus the on screen text entry function is available and opens adjacent to the text box, where it initially opened than a the bottom of the screen. This functionality is now standard in Microsoft Windows Vista.

One of our goals with InkKit was to integrate writing and drawing into one mode to counter the problems experienced with this in Freeform. This has required a great deal of work on the recognition engine as the Microsoft divider performed poorly[8]. While we continue to work on the technical side of the problem we now have a modeless writing/drawing space which has improved usability.

Display space was critical on the Tablet PCs as they have small screens and the Tablet OS did not work with any of the digital whiteboards. We ran a number of small experiments trying different combinations of screen displays and functional support for drawing. Our solution is to support two screens – the tablet for drawing on the sketch using a pen and another screen for showing the portfolio (Figure 4 – this could be a larger projected image). This is achieved by using the build in dual display modes, there both views on a single tablet screen (as standard windows) is also supported for mobile use.

We spent some time surveying people on suitable names for the different views before settling on portfolio and sketch. After various trials we clearly delineated the functionality between the views. On a sketch the user can draw and edit the ink and run the recognition engine against that sketch. In the portfolio the ink is not accessible, here the user can move and resize the sketches, draw connections between sketches and run the recognition engine against all sketches taking into account interconnectivity.

The multi-modal nature of this interface has presented some interaction problems that are not easily solved: the user needs the pen to work with the sketches on the tablet screen but the mouse to work with the portfolio when it is shown on the alternative screen. The device switching is an inconvenience. Furthermore the operating system has only one pen/mouse cursor so there is contention for this resource. The new generation of touch screen whiteboards (e.g. http://www.nextwindow.com/) and Vista OS may offer better solutions; we plan to investigate these alternatives shortly.

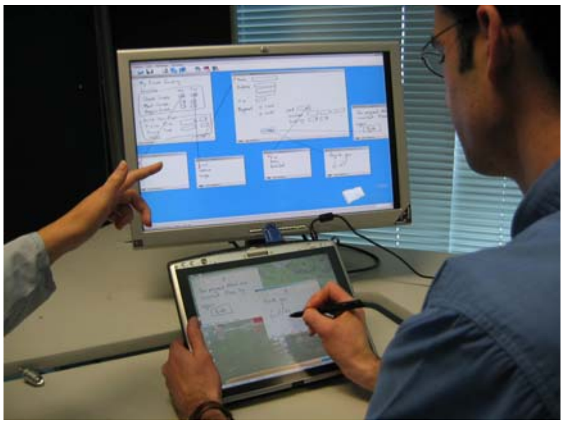
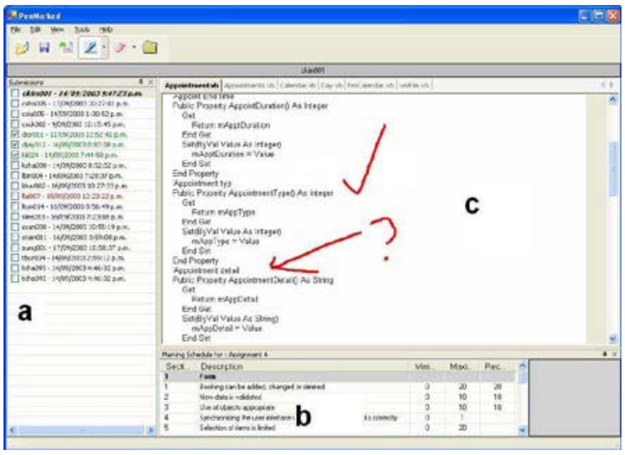


Figure 4: InkKit User Interface

4. PENMARKED

Penmarked is an annotation tool designed for marking student’s assignments [5, 12, 13]. While any type of assignment can be marked using Penmarked there is specific functionality to support marking computer programs. Our goal with this tool is a paperless environment where rich ink annotation of the student’s work is afforded along side full task support for a class set of assignments. The main interface of Penmarked (Figure 5) consists of three panes; student list, annotation pane and grade rubric. There are also a number of icons to access frequently used functions.

Penmarked was our first Tablet PC application. We conducted a number of small, informal usability tests during development and a larger formal usability test towards the end of development.



```
Public Property AppointDuration() As Integer
    Get
        Return mApptDuration
    End Get
    Set(ByVal Value As Integer)
        mApptDuration = Value
    End Set
End Property
'Appointment typ
Public Property AppointmentType() As Integer
    Get
        Return mAppType
    End Get
    Set(ByVal Value As Integer)
        mAppType = Value
    End Set
End Property
'Appointment detail
Public Property AppointmentDetail() As String
    Get
        Return mAppDetail
    End Get
    Set(ByVal Value As String)
        mAppDetail = Value
    End Set
```

Marking Schedule for : Assignment 4

Sect.	Description		Mini.	Maxi.	Rec
1	Form				
1	Booking can be added, changed or deleted		0	20	20
2	New data is validated		0	10	10
3	Use of objects appropriate		0	10	10
4	Synchronising the user interface	its correctly	0	1	
5	Selection of items is limited		0	20	

Figure 5: A screenshot of Penmarked showing the student list (a), the mark schedule (b) and the annotation frame (c).

The informal tests were particularly important to us when deciding how to configure the grade rubric to support bi-modal input (either the or keyboard). We found that writing on a screen accurately required space: to provide enough space directly in the table would have required about half the available screen space. We experimented with the built in on screen keyboard control, however users found it difficult to use and it obscured a sizable portion of the screen. We also tried an onscreen number pad (like a calculator pad), and the writing entry box visible in the bottom right corner of Figure 5.

Users found the writing entry box quicker and easier than the calculator pad. We implemented this, supported by the Tablet OS recognition engine with the factoid for numeric data (limits recognition to digits and mathematical symbols) and range checked the data against the minimum and maximum values. Valid data is automatically saved into the selected cell in the rubric if the user changes the cell selection or after a 500ms time delay. If invalid data is detected the writing box flashes red and an alert is sounded. Alternatively the user can enter marks in the rubric using the keyboard (the same validity checks are conducted). The process for arriving at this solution was a series of informal evaluations and explorations of alternatives.

The formal usability test was conducted with teaching assistants (TAs) marking real programming assignments. Five TAs spend an hour marking assignments. Think-aloud protocol with an observer and video was used. The programming assignments were small .net programs for a simple windows form.

We found that persuading the markers to talk was extremely difficult; much worse than anticipated. They appeared to be concentrating on the task and seemed to find talking a distraction. Often when they did speak, it was about the program they were marking, not Penmarked We attribute this to cognitive overload due to the cognitive demands of program review [19].

This study exposed a number of usability problems. Most significant points were noted during the first half hour of each observation rather than post analysis. Of most interest in this context were the input modalities. Penmarked was designed to support pen-only input. However the programs that the TAs were marking were not designed for the tablet. We set the study room up with the Tablet PC without a keyboard. However the TAs wanted to run the students' programs and enter data into them. Doing this with the pen and on-screen keyboard was too slow. During the first session we added a keyboard to the setup. The TAs tended to use the keyboard for text entry to the student programs and the marking rubric – often with their non-writing (left) hand and use the pen in their writing hand for annotating and mouse actions. In a later focus group discussion about this software users told us that they found a keyboard more convenient, but used the software without a keyboard if they were mobile (e.g. on the bus).

5. HAPTIC PENS

The goal of this software is to assist visually impaired children to learn to sign their names [9, 10]. Visually impaired people need to have a repeatable signature for legal documents. However it is very difficult for them to learn to write as they can not see the letters to copy or watch physical demonstrations, nor do they get any visual feedback from their own writing efforts. The system we devised links a Tablet PC, a Phantom Omni haptic pen [1] and a tactile drawing surface (special plastic on a rubber mat Figure 7). This project has been very challenging both from a design and usability test perspective.

It is a collaborative environment with a teacher and student working together: as the teacher writes on the tablet the tablet pen path is echoed on the phantom pen that the visually impaired student is holding. The phantom pen scores a line on the tactile drawing surface that the student can feel. The phantom user can write independently by holding the barrel button down. In this mode the phantom ink is echoed onto the tablet screen. We also implemented a mode where the tablet pen stroke is converted into a virtual stencil. In this mode the phantom pen is constrained to the stencil to guide the user along the stroke. The system also has two audio outputs, voice output of recognized characters and a sinusoidal tone generated from the tablet stroke that varies the audio pan and pitch to represent the user's current horizontal and vertical position on the page respectively.

Therefore this system has two users and multiple modalities – two pens, a tablet screen, a tactile surface (non computer) and two types of audio.

Usability testing this system presented some unique issues because of its multi-user, multi-modal nature and the target user group. The development team is experienced in working with visually impaired; however this is not the same as being visually impaired. To a greater extent than any other system in which I have been involved, we need representative users to help us with the development. However, there are very few visually impaired children and there are many researchers wanting access to them. We were very reluctant to ask visually impaired children (and their parents) to be involved in the early usability testing of the system. We recruited visually impaired adults to be our usability test participants.

We concentrated our testing on the visually impaired users' interface. Our first VIP user had an informal demo of the system while we were in the early stages of development. We set it up with a book for the writing surface as the phantom pen is designed to work above the desk top (Figure 6). She tried to detect the pen path on the paper – and she could, but it was difficult, hence our change to the drawing board (Figure 7).

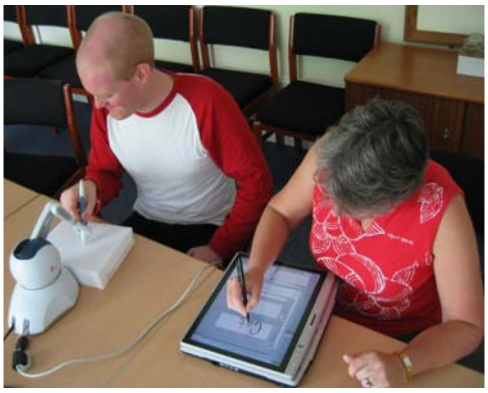


Figure 6: First Signature Setup

Four visually impaired adults were recruited for the formal usability testing. We used Morae™ to record the sessions and independently logged all the pen data points from both pens. Our plan was to first familiarize the users with the device and then to progressively test, train and re-test them on lower case letters grouped according to shape. Our pilot test indicated that this was too ambitious a goal. We reduced the task list to letters similar to 'o' – 'o, c, a, d, e'.

There were interesting issues around the tactile output. The users want this space to support their spatial orientation. Following a process of trial and error with the first two users we found it worked best if they marked the perimeter of the drawing space by tracing around it with the pen (Figure 7). They could then orient themselves within the space. As a section of the plastic became crowded we replaced it and got them to retrace the area Figure 8.

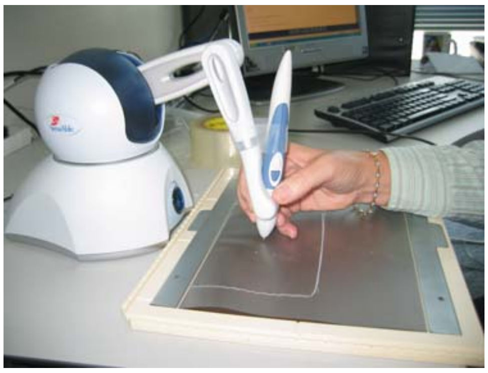


Figure 7: Signature Drawing Area

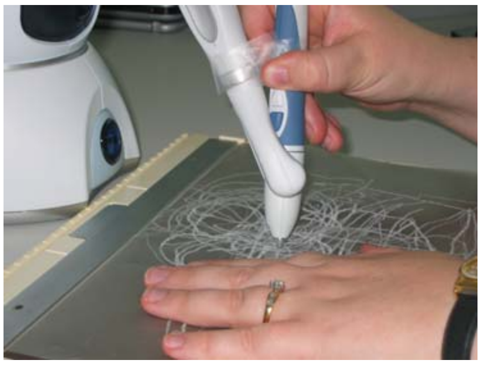


Figure 8: Signature User Experience

The phantom pen is much larger than a standard pen and it is attached to a robotic arm. Initially we did not specifically train the users on how to hold the pen. However, considering the constraints of the device and visually impaired peoples unfamiliarity with holding pens we changed our approach to include specific training on how to hold the pen. There are two barrel buttons on the phantom pen. During our first demo session we found that, sighted as well as, our visually impaired user had trouble using the barrel buttons. We minimized the use of the barrel buttons, either or both having the same effect and we use them only to record the phantom user writing independently. They are still difficult to use and we are considering other approaches to detecting when the Phantom user is writing.

The audio feedback from the character recognition was not a success because of the high error rates. We use the Tablet OS recognition engine which is more reliable recognizing words than single characters. The study participants all use standard computers with voice output for their every day computing. Having untrustworthy voice output destroyed their confidence in their writing ability. In contrast the sound feedback was helpful to two of the users who were having difficulty detecting the change in height of the pen as it was pull down to the writing surface at the start of a stroke.

The stencil mode was not useful. The bounds of the stencil were not clear to the Phantom user and it did not support the tight collaboration between the two users that worked well in the other modes. It may be useful for visually impaired user to work alone, but at this point we are not planning use it.

While most of the development and usability testing has concentrated on the phantom interface, we reviewed the tablet interface after the usability test. Initially it had two visualization spaces for ink, one for the tablet user and the other for the phantom. With our decision to demote the stencil mode we removed one of the visualization spaces as only one pen is actively creating ink at any time (Figure 9).

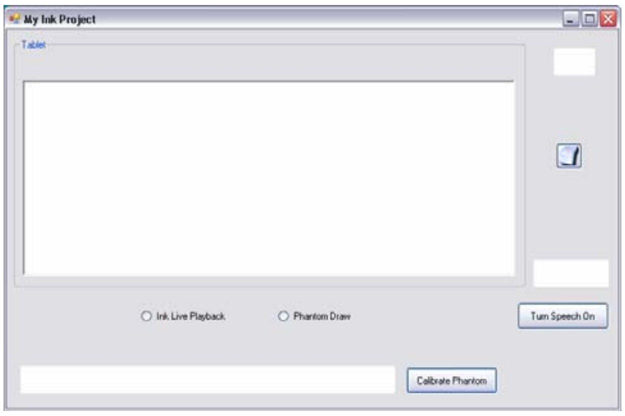


Figure 9: Signature Tablet Interface

6. DISCUSSION

Here I have presented our experiences with four different software tools where a pen is used as one input device. In each case the pen input alone has been insufficient for efficient interaction. Usability testing has been an important element of the success of each of these projects. We have used a variety of formal and informal usability testing techniques. The informal techniques have often been integrated with the implementation phase as a method of exploring alternative approaches.

As with many multimodal systems, pen-based interaction is in its infancy. We have often found ourselves in the situation of needing to make decisions about how to implement a particular feature. There is a small, but growing corpus of research on usability of pen-dominant interaction. However, often our projects have required unique solutions. Our approach has been to brainstorm alternative approaches and then build small test beds to evaluate them. The evaluations have usually been informal. Clearly this is not scientifically rigorous, however when there is little existing transferable knowledge and many decisions that need to be made it is a practical approach that has been relatively successful.

Each of the projects presented here has had at least one formal usability test. For these tests we have observed, videoed and, except for the Penmarked project, preserved screen captures. In some we have also recorded supplementary pen data. Two different specific usability testing setups have been used; a custom build usability lab at the University of Waikato and Morae ™. Neither of these setups have had the capability of preserving or analyzing pen data as opposed to mouse data.

Observation and post-task discussions with the users have, for us, been the most effective usability testing techniques. At times we have reviewed the video tapes of sessions to check particular points. In general this has been when we have noticed something of interest later in a set of studies and then wondered whether the same thing occurred with earlier participants. The collected ink data has been used to analyze and improve the recognition rates.

There are many assumptions we have made as we designed and usability tested these systems. Our experience suggests that there are a range of basic usability problems with pen based systems: parallax errors on the screens mean we must design larger interaction points (buttons etc); the barrel buttons on the pens have proved difficult to use; the nature of pen input implies the use of recognition engines and the recognition results are often inaccurate; and standard system controls may be unsuitable for pen input thus requiring either special controls be developed or the use of keyboard and mouse. Each of these usability problems require further investigation to design better fundamental solutions or interaction techniques to minimize the impact of the technology limitations.

The multi-modal nature of these systems has added complexity to both the design and evaluation of the systems. In addition we are cognizant of the effects of the particular environments which we have conducted the usability studies. This is a particular problem with mobile devices where the laboratory studies may not be representative of mobile use.

7. CONCLUSIONS

The major usability challenges with multi-modal interfaces are two fold. First, the exponential increase in combinations of devices as our experience has been that adding one interaction medium does not remove existing mediums. Second, many of the additional input modalities are in the form of continuous data (such as digital ink, speech) that must be interpreted and consistently accurate recognition is and outstanding problem.

ACKNOWLEDGMENTS

I would like to acknowledge the contributions of the various students and usability test participants that have contributed to these project.

8. REFERENCES

[1] Sensible Technologies, http://www.sensable.com/haptic-phantom-omni.htm, accessed on 8 June 2007

[2] Chung, R., Mirica, P., Plimmer, B., InkKit: A Generic Design Tool for the Tablet PC, in proc CHINZ 05, ACM, (2005), 29-30

[3] Freeman, I., Plimmer, B., Connector Semantics for Sketched Diagram Recognition, in proc AUIC, ACM, (2007), 71-78

[4] Jarrett, R., Su, P., Building Tablet PC applications, Microsoft Press, (2003),

[5] Mason, P., Plimmer, B., A Critical Comparison of Usability Testing Methodologies, in proc NACCQ, NACCQ, (2005), 255-258

[6] Mimio, Mimio, http://www.virtualink.com, accessed on 12 June 2001

[7] Nielsen, J., Usability Engineering, Morgan Kaufmann, (1994),

[8] Patel, R., Plimmer, B., Grundy, J., Ihaka, R., Ink Features for Diagram Recognition, Sketch Based Interfaces and Modeling IEEE, (2007),

[9] Plimmer, B., Crossan, A., Brewster, S., Computer Supported Non-visual signature training, in proc First International Workshop on Haptic and Audio Interaction Design, (2006), 1-4

[10] Plimmer, B., Crossan, A., Brewster, S., Patel, R., Designing a Haptic Interface for Teaching Visually Impaired Children Writing, in proc OZCHI 2007, (2007), in press

[11] Plimmer, B., Freeman, I., A Toolkit Approach to Sketched Diagram Recognition, in proc HCI, eWiC, (2007), in press

[12] Plimmer, B., Mason, P., Designing an Environment for Annotating and Grading Student Assignments, in proc OZCHI, (2004), 45-53

[13] Plimmer, B., Mason, P., A Pen-based Paperless Environment for Annotating and Marking Student Assignments, in proc AUIC, CRPIT, (2006), 27-34

[14] Plimmer, B. E., Apperley, M., Evaluating a Sketch Environment for Novice Programmers, in proc SIGCHI, ACM, (2003), 1018-1019

[15] Plimmer, B. E., Apperley, M., Freeform: A Tool for Sketching Form Designs, in proc BHCI, (2003), 183-186

[16] Plimmer, B. E., Apperley, M., Freeform: An informal environment for interface prototyping, in proc CHINZ, (2002), 11-12

[17] Plimmer, B. E., Apperley, M., INTERACTING with sketched interface designs: an evaluation study., in proc SigChi 2004, ACM, (2004), 1337-1340

[18] Plimmer, B. E., Apperley, M., Software for Students to Sketch Interface Designs, in proc Interact, (2003), 73-80

[19] Robins, A., Rountree, J., Rountree, N., Learning and teaching programming: A review and discussion, Computer Science Education, 13, 2, (2003), 137-172

[20] Rubine, D., Specifying gestures by example, in proc Proceedings of Siggraph '91, ACM, (1991), 329-337