

# CrossEd: Novel Interaction for Pen-Based Systems

Grecia Lapizco-Encinas  
University of Maryland  
College Park, MD 20742  
glapizco@cs.umd.edu

Alejandro Rodríguez  
University of Maryland  
College Park, MD 20742  
alejandr@cs.umd.edu

## ABSTRACT

We propose an interaction technique for pen-based systems based on the goal-crossing paradigm. We describe the design of CrossEd (a text-formatting tool) whose interface components are based entirely on this paradigm. We anticipate that the experience obtained from designing and implementing this tool could lead to a more natural interaction in pen-based systems.

## Keywords

pen-based systems, crossing-based interfaces, interaction design, novel interaction methods

## 1. INTRODUCTION

The popularity of pen-based systems (incorporating a small touch-sensitive screen) has emerged as an important access technology, pen-based input is well suited for accessing information in mobile computing situations and usually small pen-based computers are used by people who work away from a desk.

In these systems, the main user interface issue centers around using a pen to interact with the computer. This presents many challenges and opportunities for innovative pen-based user interfaces. However, there exists the temptation of simply extending existing mouse-based desktop interfaces. While this is the simplest thing to do, it is not clear that mouse-based interfaces (point-and-click) are appropriate for such systems. Moreover, the natural interaction that users tend to perform with a pen is in most cases under-exploited when the user is forced to use the pen as a mouse.

Previous research has explored the interaction with pen-based devices [16, 14, 17, 15], although the main interaction technique used on these systems focused on tapping a target, which is to say point-and-click. Some attempts have been made to create alternative interaction techniques, such as gestures and marking [11, 10, 4], but pointing-and-clicking

remains as the most widely used method.

However, different paradigms do exist and have been proposed and used to a limited extent. Among these, Accot[3] introduces a different style of interaction, which could be particularly useful for pen-based interfaces: goal-crossing. In the goal-crossing paradigm, actions are triggered when a certain goal or target is crossed (the cursor passes it from one side to the other) as opposed as when it is clicked. When targets consist of straight lines, the act of crossing a goal becomes the act of intersecting the line with the cursor or tip of the pen.

Studies [1, 2] have demonstrated that goal-crossing is at least as accurate and probably faster than pointing-and-clicking, but little practical work has been done to exploit the potential of goal-crossing as an alternative technique that can complement or substitute point-and-click.

In this paper we provide evidence of how goal-crossing can be used as the main interaction technique to build pen-based user interfaces. We do this by developing a text-formatting tool whose interface is entirely based on the goal-crossing paradigm.

Our application, CrossEd, provides a test bed to show how actions usually performed with a mouse (activating a button, selecting from a menu, etc), can be translated, and possibly improved, to an equivalent action in the goal-crossing paradigm. Even further, we explore how these transformations can be combined together to create a new and different way of interaction for a conventional application, which could be extended for creating other systems based on goal-crossing interfaces.

## 2. RELATED WORK

In the past we have seen interfaces specifically designed for pen-based devices. PenPoint [7] was a project that included the design of hardware, systems software, and initial applications for mobile, pen-based computers. The PenPoint operating system was based on a notebook metaphor, and the use of the pen as the primary (for most uses the only) input device. It was nearly entirely gesture-based.

In GEdit [13] design, marking and direct manipulation are combined. GEdit implemented a simple drawing application that uses exclusively a stylus for the interactions with the user. This system takes advantage of previous ideas like pie

menus [9], and exploits thoroughly the use of gestures for making selections and specifying operations. It illustrates how markings and direct manipulation can be combined together within the same interaction to perform commands where parameters can be specified within the single action.

A number of pop-up command techniques have been developed for pen-based systems, like pie menus [9] and FlowMenus [8]. Pie menus use radial selection, which enables the user to learn directional gestures so selection can be made without needing to look at the menu items. Moreover, if a gesture is made immediately upon menu activation, the menu is not even drawn. Pie menus can be hierarchical, with the end of one stroke becoming the beginning point for a secondary selection. FlowMenus use the radial geometry of Pie menus and provide multi-level of menus by using a "return-to-center" activation, thus allowing submenus to appear at the same position of their parent menus.

Interactions similar to goal-crossing have been implemented in some systems, like selecting a set of hyper-links in a web browser by crossing each of them in a single step has been demonstrated in the "Elastic Window" prototype [12], or selecting/marketing a row of emails by continuously crossing them (Lotus Notes ©).

### 3. DESIGN GOALS

Our goal is to develop a new visual language [6] for interaction with pen-based system, which is based in the goal-crossing paradigm. Winograd et. al [18] proposes a similar approach, but its focus is on large-displays instead of conventional devices. Our design is intended for mobile pen-based devices like the Tablet PC. The user interface should allow a natural and fluid interaction and be strong enough to built new interfaces for traditional applications.

We have devised the following key points to the design of CrossEd:

#### 3.1 Footprint

In traditional interfaces the user is presented with multiple information at a time, which means that several graphical objects are available for her at once, also the arrangement of these graphical objects (layout of the interface) has been demonstrated to be of significant importance. The visual elements should be designed to allow the kind of layouts needed in traditional applications, for example the new interface components should use a screen space similar to the one used by their point-and-click counterpart.

#### 3.2 Composition

One way to achieve fluid interaction with the interface is by exploiting the ability of specifying multiple commands in one stroke (e.g. all the interaction can be performed without lifting the pen from the tablet).

Repetitive tasks, like scrolling up or down in a document, or multiples searches of a word in a document, could be specially suited for this kind of composition.

A potential strength of goal-crossing interaction yields in its ability of integrating sequence of tasks in one unit. The

ability to compose a sequence of tasks would provide an advantage over pointing-and-clicking interfaces where each part of the task involves a separate interaction. For example, the execution of two actions, would involve pointing-and-clicking to two separate targets in a sequential way.

### 3.3 Richer set of interactions

Accot [3] has mentioned that crossing targets allows a richer semantic, in the sense that while a button can be pressed in a single way, a line can be crossed in several ways (left-to-right, right-to-left, etc), and each of these interactions may trigger different commands.

### 3.4 Efficiency

Our interface should be as least as efficient as the traditional interfaces based on pointing-and-clicking. If using this new technique of interaction creates an additional burden on the user, either on time or on effort, the value of the interaction technique would be diminished.

## 4. GOAL-CROSSING PARADIGM

Traditionally, interfaces that have been designed with the mouse as an interaction method are based in the point-and-click paradigm. In this paradigm, an area of the screen is acquired by the mouse by going to that specific location (pointing) and then clicking the mouse, this operation is called "target acquisition". This process constitutes the basic operation for that type of interfaces, and a set of visual artifacts has been developed and standardized to be used in that such way, like buttons, scrollbars, menus, etc.

Rather than pointing-and-clicking, in the goal-crossing paradigm, actions are triggered by crossing the boundaries of an object with the cursor. This could be seen as entering and leaving the object, or simply intersecting the object in the case of one-dimensional shapes, like lines.



Figure 1: Crossing a goal.

While formal studies has been carried on to analyze the dynamics of point-and-click interactions, and robust results like the Fitt's law are well known, it hasn't been until the recent work of Accot and Zhai [1] that a comparable study of goal-crossing interactions was performed. They present an experimental study and determine that goal-crossing tasks follow a similar relationship than the tapping task (which is the classic task for measuring user performance in point-and-click interfaces). Thus, goal-crossing shows a behavior similar to the Fitt's law, where a linear relationship exists between the time needed to complete the task and the difficulty of it, which increases with the distance between the targets and decreases with the size of them.

In a later study [3] they compared the time required to complete a task using goal-crossing and point-and-click paradigms. They show that the time needed for complete the

goal-crossing task was less or at least the same than the point-and-click with the same index of difficulty, and had error rates close to and lower than those of point-and-click.

## 5. SYSTEM OVERVIEW

CrossEd is a simple text formatting tool whose design tries to exploits the advantages of the paradigm, therefore its interface is completely based on crossing. It is intended for tasks such as proof reading, format setting (font setting, paragraph setting) and minor text modifications like cutting and moving paragraphs. These tasks require little to no text input, therefore the system does not address the problem of text input. However, if some text input is needed, it can be performed using a regular keyboard or any other method available in the particular device and platform being used.

CrossEd is designed to be used mainly with pen-based devices, such as Tablet PCs, nevertheless it can be used in traditional PC's with the use of a tablet or even a mouse.

One of the main guidelines in the design of CrossEd was to verify that all the interactions in traditional interfaces currently performed through point-and-click, can be achieved by using crossing, and even more, find points where goal-crossing interfaces outperform or result more suitable than traditional interfaces.

CrossEd provides the basic functionality of a text formatting tool. Selecting a text formatting application has the advantages of providing a set of typical characteristics and interactions exhibited in many other common applications, which allow some of the results to be generalized directly. Also, being text editors so well-known it is easy to compare the design presented in here with actual implementations of text editors based on more traditional interfaces. Finally, a text editor has the complexity necessary to provide a fertile ground for exploiting the benefits of this particular kind of interaction, and at the same time it is simple enough to avoid implementation details irrelevant to the focus of this work.

### 5.1 Alice in Crossing-land

In order to present the system let us take real scenario. Imagine that Alice, just took the Metro in her way to work, and she would like to check that everything is all right with the document that she was typing last night. At this point Alice takes out her Tablet PC, gets the stylus out and open CrossEd.

The first thing she sees is the main window (Figure 2). It does not have any visible menus, buttons or icons. As a design goal, CrossEd was intended to use direct manipulation whenever possible, so actions are triggered through gestures or menu selections performed directly on the text editing area.

### 5.2 Executing actions

In order to open her document, she first needs to access the file menu. So she presses the pen against the screen and a FlowMenu appears right under the tip of the pen.

To select an option, she performs a gesture (with the tip of the pen touching the screen) moving the pen out of the menu

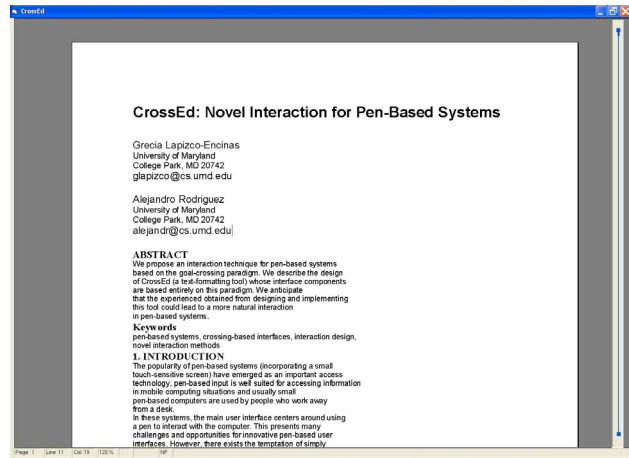


Figure 2: CrossEd main window

and then back in, as seen in Figure 3(a). When she crosses back the line of the option she wants (in this case 'File'), the FlowMenu changes to display the submenu selected. Since the pen is again inside the FlowMenu, Alice can cross the following option with a similar gesture (Figure 3(b)).



(a) Selecting an option of a FlowMenu

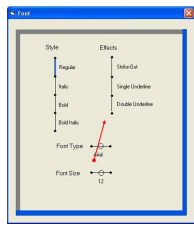
(b) Stroke for selecting an option of a submenu

Figure 3: Using the FlowMenu

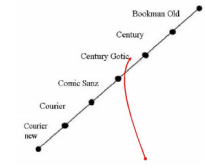
### 5.3 Exploring the file system

After selecting the corresponding option in the menu, the dialog for opening a file pops up. The dialogs display the the directory by default, in this case is "My Documents" folder, and Alice remembers that her document is in "My Documents\Customers\Apr2003". The TreeLine in the left of the dialog (Figure 4(a)) shows the files and folders in the current directory (My Documents). However, the number of files is too large to be displayed at once. She starts by crossing the TreeLine from right-to-left; now, as she moves the pen down, the files scroll down also, so she can see whole list of files. When she is on the Customers directory, she crosses from left-to-right to select it (Figure 4(b)). Then, the TreeLine to the right becomes activated and displays the files in Customers (as displayed in 4(c)), Alice crosses it from left-to-right, scroll as needed, and re-crosses from right-to-left to select Apr2003, which causes the files on this directory to be displayed on the left TreeLine which now she can cross as before. Alice could continue this zigzag interaction (without lifting the pen) until she finds the file she needs, at this point she remembers the file was on "My Documents\Customers\March2003". In order to go up one (or any number of levels), she can use the horizontal Tree-Line at the top, which displays the currently selected path at any moment. The top TreeLine in the same fashion as the

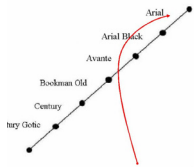




(a) Crossing the List Trigger



(b) Performing cross-and-drag to select an option



(c) Performing cross-and-drag to select an option



(d) Making the selection

Figure 7: Interaction with list of options

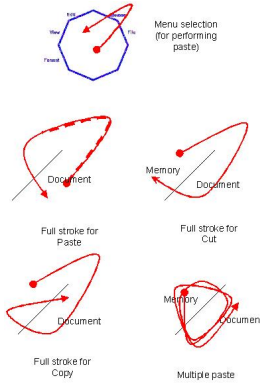


Figure 8: Cut, copy and paste

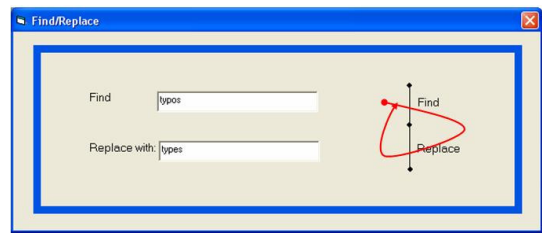
## 5.8 Find and Replace

Finally, Alice is almost done retouching her document when notices that she has used the word 'typos' in some cases she intended to write 'types'. Since it seems that she has done so more than once, she invokes the Find-and-Replace dialog.

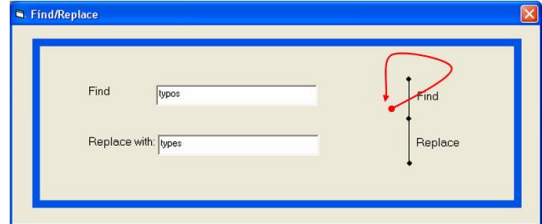
The find field is selected by default, therefore she simply uses the handwriting recognition tool of her Tablet PC and writes 'typos'. Now she crosses the replace with field to start writing on it and write (with the handwriting tool) 'types'. She has to look at all the occurrences one by one to determine whether or not to replace it, so she crosses the Find line to see the first occurrence. If she wants to replace it, she can just cross Replace (as depicted in Figure 9(a)), if not, she can "draw" a circle with her pen to cross Find again (Figure 9(b)).

Notice that she can keep doing this indefinitely without lifting the pen or even looking at the Find and Replace lines.

After the replacing, Alice takes a final look at the document and saves it using the FlowMenu. She is glad her document



(a) Performing multiples Find. Note how the end of the stroke leaves the pen in position for selecting Find again.



(b) Performing Find and then Replace. Note how the end of the stroke leaves the pen in position for selecting Find again.

Figure 9: Interaction with the Find/Replace Dialog

is looking good so she can relax for the rest of the trip.

## 6. DISCUSSION

The described application shows some of the advantages and potentialities of the goal-crossing interaction technique. Through the development of this application, it has been verified that all the interactions supported by point-and-click can be performed using crossing. Furthermore, every single visual element of point-and-click interfaces can be "translated" into an equivalent element in the visual language of crossing.

### 6.1 Mapping from Point-and-click to Goal-crossing

For the system presented in this paper, we have developed a set of interface components, which are general enough to be used in other systems. Table 10 shows the mapping between these components and the components conventionally used on point-and-click interface. While this mapping provides only a subset of all the components used on point-and-click interfaces, it supplies a wide functionality which shows how the widgets from traditional interfaces can be translated into goal-crossing.

- **Action Bars**

As opposed to conventional buttons, Action Bars can be triggered in sequences with one single stroke. While a point-and-click per button is required for every button to be activated, a single stroke might cross and activate many Action Bars, allowing for the execution of complex commands in one interaction.

One particular drawback of Action Bars is the fact that they are activated as soon as they are crossed. This could lead to unintended activation of commands by

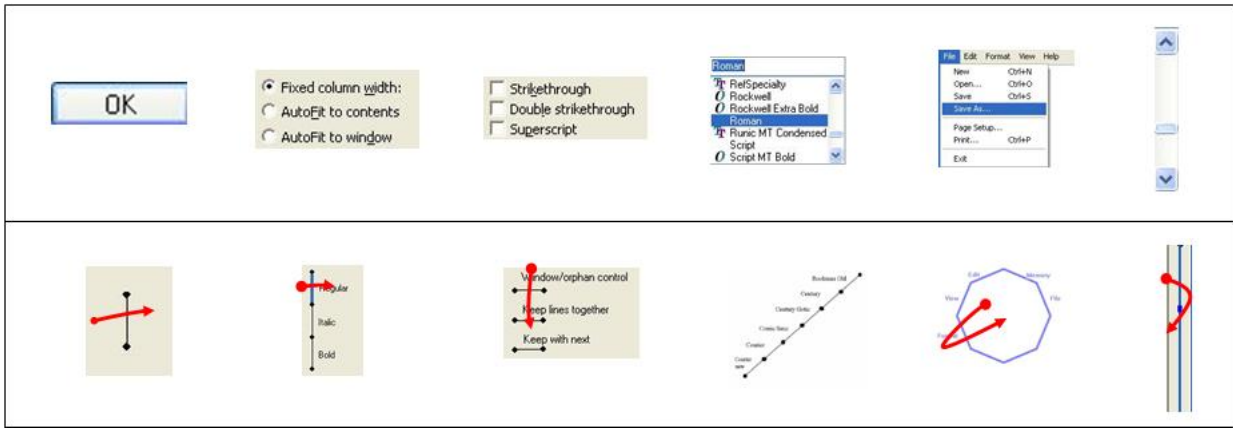


Figure 10: Mapping between components in point-and-click and goal-crossing interfaces

accidental crossing. In the case of buttons, the user can recover from accidentally pushing a button by pointing out of it before releasing the mouse. A possible solution for this problem in the case of Action Bars could be delaying the activation until the user lifts the pen, allowing the user to 'undo' the crossing by crossing in the opposite direction. However delaying the activation of commands could be confuse or hamper the composition of actions.

- **OptionLine**

OptionLines provides the same functionality of Radio Buttons adding the advantages of crossing. Since it is composed of Action Bars, it inherits the properties of these, like the fact they can be crossed in sequences with one single stroke.

- **CheckLine**

CheckLines are another clear example of how a composed action (selecting multiple options), can be performed with a single stroke. In comparison, using conventional check boxes to perform this same task would require precise pointing and repetitive actions.

- **ListLine**

ListLine replaces lists and combo boxes. One of the key features of this element is the fact that it combines exploration and selection of the list into one single interaction. This creates a bridge to gesturing which allows expert user to select any item by means of a stroke. In fact, any range of the entire list can be acceded directly by crossing the line in that particular area, which can be compared to keyboard accelerators, with the additional advantage that there is no need to switch to a keyboard or other device.

ListLine shares, though, some of the problems of traditional lists and combo boxes. In particular the usability of ListLines degrades as the total number of items in the list increases. Approaches similar to Fisheye menu[5] could be adopted to ease or solve this problem.

- **ScrollLine**

It provides the full functionality existing in a conventional scroll bar: moving to an absolute position in the document, tuning this position, scrolling a full page or line (down or up), and providing visual feedback about the current position and length of the document. However, ScrollLines eliminate the need to acquire the scroll bar thumb or target some particular spot within the scroll bar. Also, many of the actions attainable through the ScrollLine (like page up/down) are denoted by gestures. The combination of these two properties allows for blindly operating the ScrollLine, while the user keeps the focus on the document.

## 6.2 Footprint

We managed to achieve a general footprint similar to those of conventional point-and-click interfaces. Although it could be though that the nature of crossing requires more space to allow free marking and gesturing over objects, we observed that a moderate amount of space suffices for these actions to be accurately accomplished.

## 6.3 Composition

Crossing also naturally allows for the composition of gestures. This is illustrated by the ScrollLine. In this case, a single gesture, as the gesture for Page Up/Down, can be repeated arbitrarily many times to specify a repetitive action without a marked transition between repetitions of the gesture (Figure 5(a)). Moreover, gestures can be composed into a single sequence to denote, for example, a command and its arguments or modifiers. This is the case of the Repeated Page Up/Down (Figure 5(b)), where a single crossing commands the system to page up, and a second crossing, combined with the first one into a single gesture, indicates that the page up should be repeated periodically until the pen is lifted.

## 6.4 Efficiency

Crossing benefits from many factor which promises to make of the paradigm particularly efficient. Crossing draws on the user's long experience in using a pen to write or draw. Also, as it has been shown, crossing allows for continues, uninterrupted interaction without lifting the pen, making the interaction smooth and fluid and potentially fast.

## 6.5 Layout

Though it is well known the importance of proper layout in graphic user interfaces, the layout turns to be of critical importance for crossing interface. We have identified three aspects where the layout plays a crucial role in the development of the interaction (by layout we mean position, orientation and shape of the visual objects in the screen).

### 6.5.1 Position

One of the main features of crossing is that multiple actions can be triggered with a single crossing, mark or gesture, thus combining multiple commands in a single interaction. In many cases, exploiting this feature involves carefully arranging the elements on the screen so that such crossing or gesture can be easily performed. A clear example of this is the Paragraph dialog (Figure 6). Notice that the options for alignment, line space and indentation are positioned as parallel lines, which facilitates crossing them all with a single mark. This mark can be remembered by an expert user as a single gesture which she can blindly perform. Moreover, the position of the OK line allows the user, to keep crossing (no matter what selection she did) until she reaches the OK line thus activating the changes or settings.

### 6.5.2 Orientation and direction of crossing

The objects for selecting inclusive and exclusive options (from section 5.5) are an example of how the alignment of the crossing-goals can provide visual cues of their intended used. Exclusive options are presented as segments aligned in a single line. In this way, crossing (selecting) more than one option would force the user to cross in one direction, move the pen back to the other side of the line, and then cross again.

On the contrary, inclusive options are presented as parallel lines, which facilitate the user to cross them all at once with a single mark. Also, since it has to remain easy to cross a single line, the lines are tilted, maximizing the crossing area both horizontally and vertically.

### 6.5.3 Shape

the shapes of the visual objects have to be taken into account to exploit the continuous crossing feature of the paradigm and to avoid its drawbacks.

While multiple crossing is one the main advantages of the paradigm, it could also be one of its disadvantages. Since the user can cross continuously among targets, she could also cross, unintentionally, targets in the middle.

Consider for example one of the earlier designs for the List-Line, where the line is curved to optimize the space (Figure 11). The cross-and-drag interaction starts when crossing the inner arc and the selection is performed by crossing the outer arc.

As shown, a user could drag on the left side of the arc until the desired item is displayed, for example in the right side of the line. Then, instead of continuing dragging up to that point, is natural for the user to simply trace a straight line to cross the item directly. The problem arises when this crossing re-enter the drag mode by re-crossing the inner arc at a different point, which drags the list to a totally different location, making the desired item “disappear”.

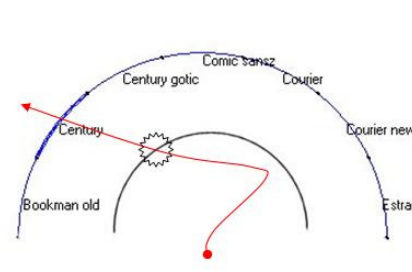


Figure 11: Radial List. Note how the shape of the widget can elicit interference

## 6.6 Cross-and-Drag

While trying to create the mentioned mapping between point-and-click and goal-crossing interfaces, we have found useful to extend the paradigm by including interactions that are not strictly crossing, for example the tree widget which responds to a continuous interaction (also the fine tuning in the scroll bar). Even though this is not pure crossing, it retains the essential properties of the paradigm and it was needed in order to achieve the required functionality.

Limiting the system to be strictly pure crossing could be too restrictive, and could lead in some cases to unnatural interactions. However there is no need to keep this restriction, since there are other interactions which blend smoothly with crossing and result natural to be performed with a pen. Among these, marking and gestures have already been explored [13]. We also found a new kind of interaction which we call *cross-and-drag*.

The ListLine presented in section 5.6 is an example of cross-and-drag where, after crossing the line, the movement of the pen drags the entire list, with the length of the list scaled to length of the line in such a way that when the pen reaches the end (start) of the line the list reaches the last (first) item.

Some of the advantages of this approach are that exploring the list and selecting an item can be combined in a single action, which can be performed without lifting the pen. Also, if the user knows roughly where the item is located, she can make the initial crossing in that general area of the line, reducing the amount of scrolling needed to find the item.

## 6.7 Occlusion

Designing interfaces based on crossing inherits some of the problems of direct manipulation and pen-based systems, like occlusion. All the visual elements of the interface have to be designed in such a way to minimize the obstruction of the visual object by the hand or pen of the user. Sometimes, this proved to be a hard task since it limits the use of screen space and constraints the direction of the interactions with the object.

To prevent occlusion, some of the elements in the system are design in such a way that the information is displayed on the left side of interaction (considering a right-handed user), thus helping to avoid the user’s hand from occluding the area of interest.

Other technique employed to ease this problem consisted on tilting some elements (as the ListLine) to convey a more natural interaction at the same time that keeping the information and the user's hand at different places.

It is to be noticed that occlusion is not a problem inherent to crossing, but an issue related to direct interfaces themselves.

## 6.8 Noise

Other issue to be considered is related to pen devices technology. Some of these devices are less accurate than traditional pointing devices like mouse, for example the pointer and the tip of the pen do not coincide on the screen. While this poses a problem for point-and-click interfaces where small buttons have to be tapped or acquired we found that this has little or no influence in the interaction with goal-crossing interfaces since the action of crossing required less precision than tapping.

## 6.9 Consistency

As with traditional interfaces we have provided a set of visual elements that are kept consistent throughout the design. In this way, the user can become familiar with those elements and recognize their functionality and interaction required.

## 7. CONCLUSIONS AND FUTURE WORK

In this paper we have explored goal-crossing as an alternative interaction paradigm for pen-based devices. We have shown, through the development of a practical application, how this paradigm can be used in replacement or as a complement to traditional paradigms to achieve more fluid, natural and accurate interaction with such devices.

We have developed a set of visual components that directly map into the components of point-and-click interfaces, and shown how to combine those to achieve a synergetic benefit not possible with point-and-click.

We have discussed several issues to be considered during the design of system based on this new paradigm, and presented plausible solutions. Our work shows that this paradigm, which has been studied from a theoretical point of view before [3], can be applied in practice to the design of real world applications, and found that goal-crossing presents an ample range of advantages making it particularly suitable for pen-based and direct manipulation systems.

In this work we have focused on providing the same functionality of traditional point-and-click interfaces while exploiting some of the blessings of goal-crossing. It remains to study the benefits of extending this functionality beyond the conventional interactions of point-and-click, in the design of essentially different, more usable graphic interfaces.

It is also apparent that goal-crossing can be more appropriate for some particular applications. These applications could be characterized and goal-crossing interfaces could be specially designed to account for the particular requirements of these applications and the particular properties of crossing that could suit them.

Finally, another important direction of research is to explore

when and how goal-crossing can be combined with the traditional point-and-click, to take advantage of the better of both.

## 8. REFERENCES

- [1] Johnny Accot and Shumin Zhai. Beyond fitts' law: models for trajectory-based hci tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 295–302. ACM Press, 1997.
- [2] Johnny Accot and Shumin Zhai. Performance evaluation of input devices in trajectory-based tasks: an application of the steering law. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 466–472. ACM Press, 1999.
- [3] Johnny Accot and Shumin Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *Proceedings of ACM CHI'2002 Conference on Human Factors in Computing Systems*, pages 73–80, 2002.
- [4] Daniel Avrahami, Scott E. Hudson, Thomas P. Moran, and Brian D. Williams. Guided gesture support in the paper pda. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 197–198. ACM Press, 2001.
- [5] Benjamin B. Bederson. Fisheye menus. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 217–225. ACM Press, 2000.
- [6] P. Bottoni, M. F. Costabile, S. Levialdi, and P. Mussio. A visual approach to hci. *ACM SIGCHI Bulletin*, 28(3):50–55, 1996.
- [7] Robert K. Carr and Dan Shafer. *The Power of PenPoint*. Addison-Wesley Longman Publishing Co., Inc., 1991.
- [8] François Guimbreti re and Terry Winograd. Flowmenu: combining command, text, and data entry. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 213–216. ACM Press, 2000.
- [9] Don Hopkins. The design and implementation of pie menus. *Dr. Dobb's Journal*, 16(12):16–26, 1991.
- [10] Allan Christian Long Jr., James A. Landay, and Lawrence A. Rowe. Implications for a gesture design tool. In *CHI*, pages 40–47, 1999.
- [11] Allan Christian Long Jr., James A. Landay, Lawrence A. Rowe, and Joseph Michiels. Visual similarity of pen gestures. In *CHI*, pages 360–367, 2000.
- [12] E. Kandogan and Ben Shneiderman. Elastic windows: A hierarchical multi-window world-wide web browser. In *ACM Symposium on User Interface Software and Technology*, pages 169–177, 1997.
- [13] Gordon Kurtenbach and William Buxton. Issues in combining marking and direct manipulation techniques. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 137–144. ACM Press, 1991.

- [14] Toshiyuki Masui. Integrating pen operations for composition by example. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 211–212. ACM Press, 1998.
- [15] Thomas P. Moran, Patrick Chiu, William van Melle, and Gordon Kurtenbach. Implicit structure for pen-based systems within a freeform interaction paradigm. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 487–494. ACM Press/Addison-Wesley Publishing Co., 1995.
- [16] Xiangshi Ren and Shinju Moriya. Improving selection performance on pen-based systems: a study of pen-based interaction for selection tasks. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(3):384–416, 2000.
- [17] John L. Sibert and Mehmet Gokturk. A finger-mounted, direct pointing device for mobile computing. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 41–42. ACM Press, 1997.
- [18] Terry Winograd and François Guimbrétière. Visual instruments for an interactive mural. In *CHI '99 extended abstracts on Human factors in computer systems*, pages 234–235. ACM Press, 1999.