FREEFORM: A TOOL FOR SKETCHING FORM DESIGNS

Beryl Plimmer
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland
New Zealand
b.plimmer@auckland.ac.nz

Mark Apperley
Department of Computer Science
University of Waikato
Private Bag 3105, Hamilton
New Zealand
m.apperley@waikato.ac.nz

ABSTRACT

This demonstration shows the tool we have developed for hand-sketching user interfaces. Our motivation for developing this tool is to provide an environment where novice programmers can move freely along the design continuum from informal low-fidelity prototypes to completed formal designs. A low-cost digital whiteboard is used to provide a shared work space for Freeform. The tool is integrated into a programming IDE and provides pen-based sketching and editing, a storyboard, run mode, recognition of shapes and words and conversion into a formal design in the programming IDE.

Keywords

Sketching, Interface Design Tools, Informal Design

1. INTRODUCTION

Hand-drawn designs have long been the preferred first rendering method for designers yet there are few computer applications that support hand-drawing. This paper provides a review of computer-based sketch tools and describes the current version of our own sketch tool. Freeform has been developed specifically for novice programmers, it runs as a Visual Basic 6© (VB6) add-in so as to provide an integrated environment for students to design their program interfaces. Section 2 describes our motivation for this project and reviews a number of other sketch tools. Section 3 describes Freeform. The evaluations we have done are described in Section 4 and Section 5 discusses possible future developments in this area.

2. BACKGROUND

Designers from a wide range of disciplines choose to use pen and paper or whiteboard for there first rendering of designs [7]. Such informal tools have a distinct advantage during the early stages of design as they require minimal cognitive effort and do not constrain design decisions to a prescribed set of widgets. Computer based design tools have been shown to overly constrained designers [7].

A number of computer-based informal drawing tools have been developed. Southerland [14] propose pen input in 1963, his work was based on cathode-ray displays. In the early 1990's Xerox Parc explored physically separated shared design spaces [3]. Xerox Parc have also developed liveboard [6] a digitised whiteboard and explored its use with meeting support software [11], [10].

Landay and his associates have developed two sketch based design tools. Silk [8] is a form design tool that takes input from Wacom tablets. There is a design space where a number of widgets are recognised using Rubine's algorithm [13] for stroke recognition and rules are used to combine ink strokes. On the storyboard navigation links can be drawn between forms that can then be used in run mode. The run mode also animates a number of widgets for example scroll bars can be dragged. The sketches can be exported into VB5 and Garner User Interface Development Environment formats.

Landay's team have also developed Denim [9] for Web site design. It expands the storyboard of Silk to have five levels of zoom to provide better overview of a site hierarchy and navigation. HTML code can be generated from Denim.

Knight is a UML CASE design tool developed by Damm et al. [4, 5] that uses Smartboard, a commercially available electronic whiteboard. This tool also uses Rubine's algorithm [13] for basic shape recognition and designs can be converted into WithCase diagrams. Damm et al. support different levels of formality and informality within the one diagram. This tool also includes a radar window to aid navigation around large design spaces.

Bailey and Konstan's [2] Demais is for designing multimedia applications. It supports basic sketching and a storyboard and also allows the user to include media such as pictures, video and sound clips. They have placed a greater emphasis on providing support for behaviour in run mode; the designer can attach behaviours to a widget so that a click or double click invokes actions such as playing media clips. The evaluation studies of Demais showed that

it was on a par with traditional tools for creating designs and better at demonstrating behaviour.

3. FREEFORM

Freeform tool has been developed as a VB6 add-in so that users can move freely along the continuum from informal to formal design. Two prototypes have been completed, each has been usability tested and the second was also evaluated for its usefulness as a design tool for students. Here the second prototype is described under the following sub sections; physical interface, sketch space drawing and editing, the storyboard, run mode, recognition, and transformation to a formal diagram.

3.1 Physical Interface

We have constructed a low-cost interactive digital whiteboard to provide a shared work space [1]. A standard data projector is used to project a computer screen image on to the back of an opaque glass screen that has a Mimio© digitiser bar attached. The Mimio pens are used in mouse emulation mode to supply the program with ink input. Although it is possible to emulate right-mouse actions by pressing a button on the digitiser bar, usability testing suggested this is difficult, therefore all the interaction is via the pen.

3.2 Sketch Space

The sketch space (Figure 1) endeavours to honour the whiteboard paradigm, although users need to be aware of the requirements for recognition. There are two inking modes, drawing and writing, that are differentiated by colour. Each drawing pen stroke is held as a separate glyph, writing strokes are joined together by proximity into words.

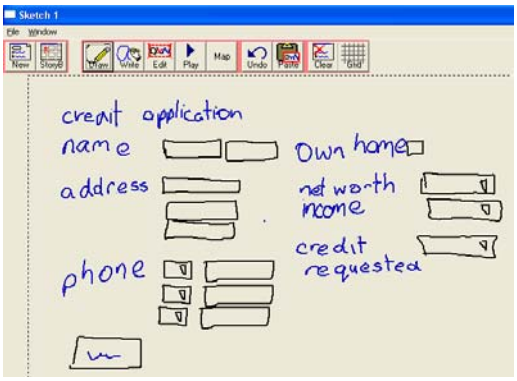

Figure 1 Sketch Space

In edit mode users can select one or a group of glyphs and then copy, move or resize the selection. Selected ink can also be changed from drawing to writing or visa versa. Editing is a multi-step process as usability testing showed that this worked best. First the user must change to edit mode then select the stroke or strokes to be changed. The selection is then highlighted with larger than normal handles on the perimeter because of pen accuracy and parallax errors cause by the glass screen. Once selected users can grab a handle to move or resize the selection or click the appropriate button for deleting, copying or changing ink modes.

Two editing functions are included as part of the basic drawing/writing gestures; a delete gesture that deletes underlying ink and if a new drawing stroke lies approximately over an old stroke the old stroke is replaced with the new. There is an infinite undo stack so that all user actions can be progressively reversed. A grid can be shown on the drawing space; the software uses this grid during the transformation process to align controls. We found many users preferred to have the grid visible while sketching as they found it easier to write and draw with guidelines. Ink colours and grid size are configurable by the user.

3.3 Storyboard

Users can create multiple forms; miniatures of these forms are shown in the storyboard view (Figure 2). In this view the user can move a form around the storyboard by dragging it to a spare slot or delete a form by dragging it to the trash can. The user can also add navigation links between forms by placing the pen down on the source spot on one form and dragging to the destination form. Navigation links can be moved or trashed and these actions can be reversed using the undo.

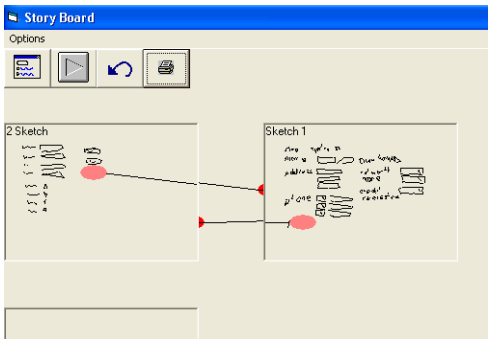

Figure 2 Storyboard

3.4 Run mode

In run mode the sketch is shown as a background layer with the navigation links activated. The user can write on the form (in a different colour) imagining that they are running the program thus checking the design and navigation between forms.

3.5 Recognition

Most sketch systems disclose recognition to users progressively; we have chosen not to do this so as to not distract the user from the design task. Our philosophy is that recognition is unimportant until the user wants to either add functionality to glyphs or convert the form into a formal design. Three recognition techniques are employed.

Two libraries of pen strokes are maintained; drawing shapes and letters. These libraries are fully exposed to the user so that they can add their own examples of specified strokes or add new classes of strokes. All pen strokes are immediately recognised using a modified Rubine's algorithm [13] against either the drawing shape or letter library. Only delete strokes result in any immediate action.

There is also a rule base for combining different drawing strokes to make VB controls. A control can be defined as a single stroke, two strokes or a container; there can be multiple definitions of a control. For single stroke controls the user simply selects the stroke from the list provided. For two stroke controls or containers the user specifies the primary stroke and secondary strokes with the relationship between the two. The second stroke maybe required or optional and there maybe more than one type of secondary stroke. If more than one secondary stroke is included they are treated as logical ors. For example a dropdown list is defined as a textbox (medium sized rectangle) that contains a small circle or square (most people draw a small triangle that is classified as either a radio button or check box).

Word recognition is lowercase characters only, it is achieved by using Rubine's algorithm to recognise strokes and then combining strokes for letters that are naturally formed with two strokes such as 't' and 'i'. At this point each letter is represented as a list of possible letters with a probability weighting. These lists of letters are then matched against a vocabulary and the most likely word selected. If the most likely word has an average letter probability of greater than 3rd place the software considers it has not matched the word.

3.6 Translation

When the user is ready to translate their sketch into a VB form they click the 'map' button on the sketch interface. The recognition algorithms described above are run and the sketch is tidied by placing each glyph onto a grid intersection point. Recognition is revealed by superimposing the type of each glyph and words, as labels onto the sketch. The user can correct any recognition errors by clicking on the label and changing the glyph type or selecting a new word from the vocabulary list.

As part of the rule base for combining strokes described above the user can also define how the VB control attributes are generated from the sketch. The program dynamically creates a list of all the attributes of the specified control and shows a list of the ink attributes. A control attribute can be created directly from a sketch attribute, for example the sketch left position can be use as the control left position. Attributes can also have fixed, minimum, maximum, or unit values, these are useful to standardise sizes and make the form look tidy.

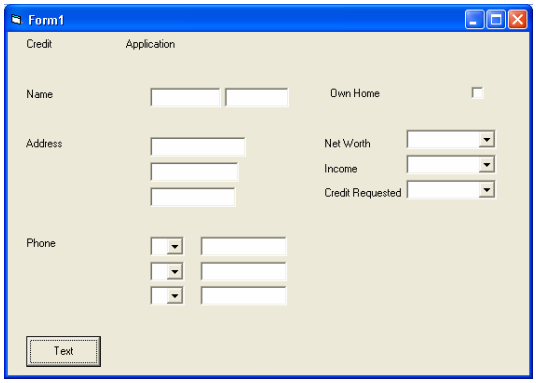

Figure 3 Translated form

Once the user has mapped the sketches and corrected any recognition errors they can generate the VB form by clicking another button (Figure 3). The user can return to the sketches at any time, alter it, and regenerate the VB form.

4. EVALUATION

The usability study conducted on the first prototype suggested that some people found the pen too large and difficult to use. We were concerned that the requirement to draw shapes in a single stroke would be difficult for users and would interrupt their design process, however they adapted to this extremely quickly and none felt that it caused them any problems.

Two alterations to editing were made between prototypes, adding an undo and changing the delete gesture. The first prototype used a horizontal zigzag as both a text holder and a delete. This confused both the users and the software. The second prototype retained the zigzag as a text holder but used continuously overlaid circles as the delete gesture.

Our first prototype did not include any character recognition and the general attitude of the students is if there is no recognition then there isn't any point writing. Attempts to interface to commercial character recognition components were unsuccessful. The word recognition in the current version is limited and unreliable; however students are surprisingly happy to work with it knowing that they can choose the correct word from a list.

The transformation process was also improved from the first to second prototype. Originally the sketch glyph raw attributes were used to generate the VB controls however this resulted in each of the controls being a different size and nothing lining up. By fixing everything to a grid and applying fixed or unit values to heights and widths of controls the VB form is much more as one would expect a formal design to be.

The second prototype is much improved however there are a number of outstanding issues to be addressed. The pen could be refined substantially and it would be useful to have a button on the pen to generate right-mouse events.

Integrating better character recognition is also an outstanding objective. When the form is tidied in preparation for translation sketch ink is moved onto the grid intersection points. In the most recent study it was clear that this was a mistake, if the software moves the ink it disrupts the users picture which we now believe it is important to maintained unaltered

We also ran a comparative evaluation study to ascertain the usefulness of this tool to novice programmers. Details of this study are reported elsewhere [12]. In summary the students enjoyed using the tool, developed a more positive attitude to sketching and created more appropriate designs for the sample problems.

5. DISCUSSION

The continued use of low-fidelity tools by designers in preference to current computer design environments suggests that they have significant advantages in the early stages of design work. A number of sketch tools have now been developed to support low-fidelity design. Reliable recognition is important for functional gestures and transformation to formal environments and there are still improvements to be made in this area. However we believe that recognition should not interrupt the design process and only be disclosed on request.

Freeform is implemented as a VB add-in; however it has been designed so that the drawing space is independent of VB and the recognition is configurable. It is possible to use these building blocks to implement a sketch interface into other programming IDEs and diagramming tools such as CASE tools.

Working with a pen directly in a public space, places quite different requirements on the software to mouse and keyboard or private space pen input. Also each domain has its own particular requirements that are becoming evident as researchers explore different domains. For example in CASE diagrams connectors between sketch elements are important and the diagrams are usually large so different navigation techniques are required.

Tablet PCs provide a new platform for pen-based design tools that we would like to explore and are also likely to increase access to programming components such as character recognition. Informal interfaces have clear advantages for early design work while current computer-based tools are better for editing and emulating functionality, sketch tools show potential to combine the best of both environments.

6. REFERENCES

[1]. Apperley, M., et al. 2001, Lightweight capture of presentations for review. in *IHM-HCI*. Lille: ACM.

[2]. Bailey, B.P. and J.A. Konstan. 2003, Are Informal Tools Better? Comparing DEMAIS, Pencil and Paper, and Authorware for Early Multimedia Design. in *CHI 2003*. Ft Lauradale: ACM.

[3]. Bly, S.A. and S.L. Minneman. 1990, Commune: A shared drawing surface. in *Conference on Office Information Systems*.

[4]. Damm, C.H., K.M. Hansen, and M. Thomsen. 2000, Tools support for cooperative object-oriented design: Gesture based modelling on and electronic whiteboard. in *Chi 2000*: ACM.

[5]. Damm, C.H., et al. 2000, Supporting Several Levels of Restriction in the UML. in *UML 2000*. York, UK.

[6]. Elrod, S., et al., (1992), Liveboard: A large interactive display supporting group meetings, presentations and remote collaboration. *CHI '92*, 1992: p. 599-607.

[7]. Goel, V., (1995) *Sketches of thought*. Cambridge, Massachusetts: The MIT Press.

[8]. Landay, J. and B. Myers, (2001), Sketching Interfaces: Toward more human interface design. *Computer*, 2001. 34(3): p. 56-64.

[9]. Lin, J., et al. 2000, Denim: Finding a tighter fit between tools and practice for web design. in *Chi 2000*: ACM.

[10]. Moran, T.P., P. Chiu, and W. van Melle. 1997, Pen-Based interaction techniques for organizing material on an electronic whiteboard. in *10th Annual Symposium on User Interface Software and Technology*. Banff, Canada: ACM SIGSOFT.

[11]. Pedersen, E.R., et al. 1993, Tivoli: An electronic whiteboard for informal workgroup meetings. in *Interchi '93*: ACM.

[12]. Plimmer, B.E. and M. Apperley. 2003, Evaluating a Sketch Environment for Novice Programmers. in *SIGCHI*. Ft Lauradale: ACM.

[13]. Rubine, D. 1991, Specifying gestures by example. in *Proceedings of Siggraph '91*: ACM.

[14]. Sutherland, I.E. 1963, Sketchpad: A man-machine graphical communication system. in *Spring joint computer conference: American Federation Information Processing Societies*. Montvale, New Jersey.