InkKit: A Generic Design Tool for the Tablet PC

Ronald Chung[1], Petrut Mirica[2], Beryl Plimmer[3]
University of Auckland
Private Bag 92019, Auckland
New Zealand
Phone: +64 9 3033670

ronaldc83@hotmail.com[1], p_m_82@hotmail.com[2], beryl@cs.auckland.ac.nz[3]

ABSTRACT

In this paper, we describe the design philosophy, implementation and evaluation of InkKit, an informal design platform that uses pen input on a tablet PC to imitate the informality of a low fidelity tool. The aim is for this toolkit to provide a foundation for further research into domain specific sketch support.

Designers initially hand-sketch their ideas [3, 6] because informal tools, such as pen and paper, offer the freedom to work with partly formed or ambiguous designs. The emergence of electronic pen input systems has seen a number of exploratory projects applying pen-based sketch software to the design process. Even though these projects differ, most of them use the same general framework. Thus a significant part of the implementation incorporates the same basic functionalities.

Categories and Subject Descriptors

H.5.2 **[User Interfaces]**: *Input devices and strategies* D.2.2 **[Design Tools and Techniques]** *User Interfaces*

Keywords

Sketch tools, toolkits

1. INTRODUCTION

InkKit intends to act as a starting point for other projects by providing an intuitive, generic and extensible sketch space. It provides the basic functionalities expected in computer aided design software with a user-friendly interface. InkKit also implements commonly used techniques such as recognition and beautification. In addition, it is designed to be extensible in the sense that it is capable of accommodating various types of designs (such as UML diagrams, ERDs and interface designs) that can integrate with different formal design environments.

2. RELATED WORK

FreeForm2 [6] is a design tool specifically implemented to convert hand-drawn sketches into Visual Basic 6 forms with input recognition and beautification. It also provides a storyboard mode that allows users to create connections between sketches. Leszynski InTegrate [4] is another sketching tool that converts user input into strictly defined form components. DENIM [5] is particularly developed for use in the early stage of website or form design. It allows storyboarding with zooming capabilities. SUMLOW [6] allows various UML diagrams to be sketched and it incrementally formalises the diagram, recognizing UML notational symbols as they are drawn. A similar project called Ideogramic UML [2] was created to transpose user gestures into solid UML objects.

3. REQUIREMENTS

In order to accomplish the goal of being intuitive, generic and extensible InkKit should meet the following requirements. First, it should provide an effective user interface which is easy to use so that the designer can freely create and manipulate sketches without interruption to the creative process. Second, InkKit should perform basic reliable text and drawing recognition. Third, it is required to handle multiple designs simultaneously and provide a way to create links between those designs. Lastly InkKit should be able to beautify abstract, informal designs in preparation for formalisation.

4. DESIGN AND IMPLEMENTATION

InkKit is implemented using Microsoft Visual Studio.NET for Windows XP Tablet PC. It runs on all Microsoft platforms that support electronic ink input, but requires the Tablet OS for recognition.

In order to satisfy the requirements presented above InkKit incorporates the following functionalities. First, it allows inking whilst supporting all the necessary functions usually contained in an editing environment such as undo & redo and save & load.

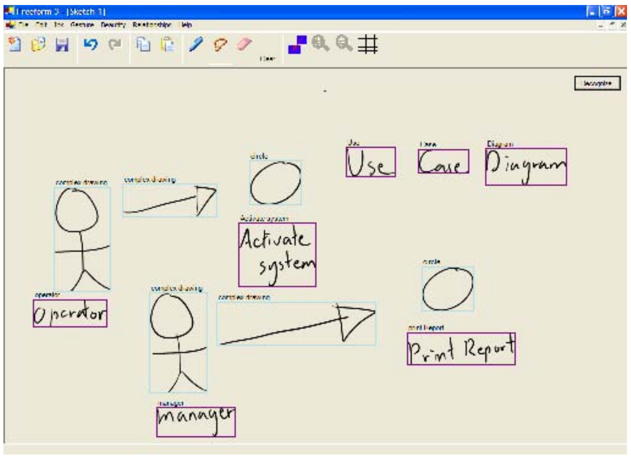

Figure 1 Recognising a use case diagram

Second, the program incorporates modeless recognition; distinguishing between words and characters, and typical simple shapes (figure 1). This is achieved by pre-processing ink to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHINZ'05, July 6–8, 2005, Auckland, NZ
Copyright 2005 ACM 1-59593-036-1/04/10

separate it into writing and drawing. The Microsoft text recogniser is then used to recognise writing and Rubine's [7] algorithm is employed for the shape recognition. We have included an interface to the gesture library so that the user can customise the gesture sets; thus InkKit can dynamically learn how to recognize new components. The last step of recognition is to apply adjacency rules to collect recognised elements into complex shapes.

Third, InkKit also supports a storyboard mode with zooming capability that allows users to easily create relationships between existing sketches (figure 2). To handle multiple designs we implemented MDI forms. While in storyboard mode the MDI forms can be linked with connectors that act as an electronic rubber band between components on the same or a different form thus allowing for complex diagram creation.

To meet the last requirement, beautification, InkKit standardises recognised components to predefined sizes based on the taxonomies specified by the user [1]. Components are snapped to the grid; and groups aligned horizontally and vertically; any overlapping that might occur during the process is automatically resolved.

To offer more flexibility, the beautification process has been divided into different steps. Thus the user can, for example, prevent overlapping of objects on a sketch without aligning them.

By default all components on a sketch are recognized and beautified as a whole. This can also happen at the selection level with slight adjustments to the algorithm.

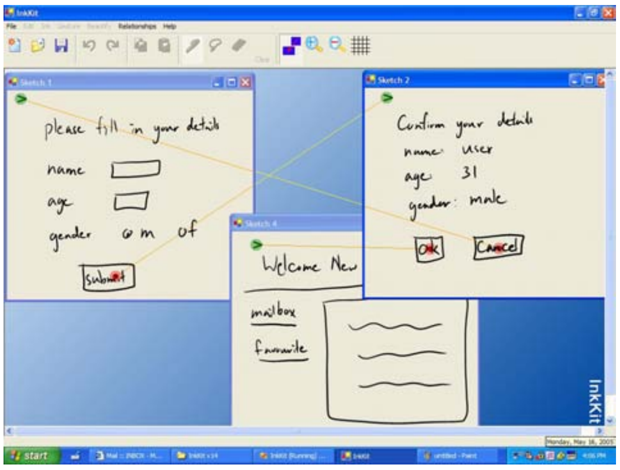

Figure 2 designing a user interface in relationship mode

5. EVALUATION

Evaluation of InkKit is divided into user interface, recognition, and beautification.

General user interface design guidelines and the experiences of others [5, 6] are followed to ensure intuitiveness. However, due to the inherent problems of pen input technologies, additional steps were taken such as addressing the parallax problem by increasing the area allowed for selecting components.

In general modeless recognition performs successfully in the sense that the strokes are reliably grouped and differentiated in most of the cases. To achieve higher accuracy user-specific training is required at the basic level, such as handwriting size. Text recognition is generally reliable since we utilised the built-in recogniser of the tablet OS. The accuracy of the recognition result for shapes relies on Rubine's [7] Algorithm, which was tested for implementation correctness.

The beautification process has been tested by considering each technique individually. Snapping to grid and standardisation of components achieved satisfactory results. Alignment and overlapping perform as expected in most situations, however may fail with complex diagrams unless more restrictive rules are imposed.

6. CONCLUSION & FUTURE WORK

Although the program can still be refined in many ways, InkKit, in its current state is meeting the requirements presented above and is ready to be used as a generic toolkit to provide the foundation for further research into domain specific sketch support.

Further improvements can be made especially to modeless recognition and beautification in order to increase accuracy. Extensive research is currently being performed in these fields and more reliable techniques can be added as they are found.

InkKit can now be extended to accommodate various design techniques. Rules can be applied to a sketch so that it can be converted into application or language specific formal design such as a webpage with automatic HTML conversion, an ERD in ERWin or a use case diagram in Rational Rose.

7. REFERENCES

[1] Blackwell, A.F. and Y. Engelhardt, A Meta-Taxonomy for Diagram Research, *in Diagrammatic Representation and Reasoning*, M. Anderson, B. Meyer, and P. Olivier, Editors. 2002, Springer.

[2] Damm, C.H. and H.R. Hansen, Ideogramic. 2002. http://www.ideogramic.com/ Accessed 10 Jan 2005

[3] Gross, M., The proverbial back of an envelope, *in IEEE Intelligent Systems.* 1998. 10-13.

[4] Leszynski Group, InTegrate. 2005, http://www.leszynski.com/ Accessed 10 Jan 2005.

[5] Newman, M.W., et al., DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. *in Human-Computer Interaction*, 2003. 18(3): 259-324.

[6] Plimmer, B.E. and J. Grundy. Beautifying sketch-based design tool content: issues and experiences. *in proc AUIC.* 2005. Newcastle: ACM: 31-38.

[7] Rubine, D. Specifying gestures by example. *in proc Proceedings of Siggraph '91*. 1991: ACM: 329-337.