Computer-Aided Sketching to Capture Preliminary Design

Beryl Plimmer
Department of Information Systems
Manukau Institute of Technology
Private Bag 94006, Auckland, New Zealand
Beryl.Plimmer@manukau.ac.nz

Mark Apperley
Department of Computer Science
Waikato University
Private Bag 3105, Hamilton
M.Apperley@cs.waikato.ac.nz

Abstract

This paper describes the vital role of freehand sketching in the design process. When designers first tackle a design problem they usually do so by sketching. We will explore the essential elements of sketching that make it so helpful to problem solving. We then examine how current computer interfaces interfere with the sketching process, and go on to establish the requirements for an environment to support sketching. Finally we describe a system under development to integrate sketching into a visual programming environment (Visual Basic).

.*Keywords*: Sketching, design, informal interfaces, pen computing, large interactive displays.

1 Introduction

Sketching is the preferred preliminary capture process for designers because it provides a quick and easy way to externalise design ideas. It is well suited to ill-structured problem solving (Goel, 1995) as participants can produce, evaluate, modify, refine and replace ideas rapidly. The requirements for a sketching medium are simple, yet few existing computer interfaces facilitate true sketching. This is because providing a sketching interface that is more useable than the more formal alternatives has been technically difficult from both a hardware and software perspective. We propose a model for a sketch interface and argue that it is now possible to provide a cost effective sketch interface with software support so that the sketch artefacts can be used as input to the next design step.

2 Sketching

Sketching is effortless and natural; we learn to sketch as very young children. The sketching process occupies almost zero cognitive load, and this is important as it allows the designer to externalise a design while directing all the cognitive effort to the design process (Goldschmidt, 1999). Design is an iterative process (Tversky, 1999); the iterations may be very quick during the early phases with designs being constructed, refined, and discarded in rapid succession. Typically, sketches are informal, abstract conceptualisations of reality using highly stylised icons and constructs.

People use a range of mediums for sketching: a stick on the sand, paper and pencil, whiteboards and pens. These surfaces share the characteristics of providing a direct, rapid and effortless way to express ideas visually (Goldschmidt, 1999). Sand is easy to alter but lacks permanency and portability. Whiteboards are easy to alter and large enough for a group to share, however the available space is generally quite small and like sand it is a non-permanent, non-portable surface. In contrast, paper provides more permanency and does not have the same sort of space restrictions. One can choose a piece of paper of appropriate size for the scale of the task and extra paper is generally readily available, but it is more difficult to alter.

Most creative fields share a tradition of sketching as an essential part of the design process. A design starts as a cognitive impression that may be quite vague, that the designer then expresses in some external form in order to work with it (Tversky, 1999). Designers tend to choose sketching for this first external presentation because it allows them to download short-term memory quickly on to a more permanent space. Architects sketch to explore spatial relationships and as a means of communicating with others. Engineers find that sketches not only allow them to explore the visual components of a design but also help expose the underlying functional requirements. Software engineers use visual modelling to describe abstractions of real world objects composed of both information and processes. User interface designers use diagrams to describe both the appearance and functionality of an interface.

The sketch serves as a cognitive support tool during the design process; it compensates for human short-term memory limitations and at the same time supplements cognitive effort by depicting the mental imagery in a concrete form. Where a detailed nontrivial design is too large to hold as a mental model, a sketch allows the designer to visually describe the overall concept and then reorganise, refine and explore the details (Goldschmidt, 1999); different levels of refinement can reside simultaneously in the same diagram. In this way an ill-structured problem slowly resolves into a structured solution. This process of partial representation makes good use of our innate visual intelligence.

Many design activities require group collaboration or agreement. This is best achieved if the group works together to produce the initial design concepts (Bekker, 1993). By working together the participants share a better understanding of the problem and solution space. A sketch sets up a visual dialogue with the designer and other group members and facilitates the identification of

patterns and relationships (Tversky, 1999). Practitioners have found that the 'polished' diagrams that computers produce discourage critical evaluation and discussion (Goel, 1995). This is so vital to the design process that a number of pieces of software have been created to transform typical computer produced diagrams into something that resembles a hand-drawn sketch. Other designers admit that they trace over their computer output with pen to present a first-draft to clients (Landay, 1996).

Hierarchical structure and sequence are also important in many design problems; these associations can be easily identified and explored with a sketch. It has also been shown that while sketching, experienced designers think about the underlying processes (Tversky, 1999).

The range of symbols that is used in sketches and diagrams is generally quite small: rectangles and ovals of different dimensions and orientation, straight, curved and squiggly lines. Glyphs (or visual symbols) may be a single symbol or a combination of symbols. Their meaning is context dependent (Gross, 1998), with each discipline developing a unique set of meaningful glyphs. For example architects have specific icons for showing doors and windows, physicists have symbols for heat loss. Software Engineers typically use a diagrammatic methodology to model systems where the methodology defines a symbol set. User interface designers use abstract placeholders and symbols similar to the visual appearance of the GUI controls. In fact, the same icon may have different meanings in different disciplines or even within the same sketch depending on context.

The majority of engineering, architectural and software diagrams ultimately end up on computers and of course the end product of a user interface design is the computer interface. However most designers start with hand-drawn sketches. They give a number of reasons for this.

First sketching is quicker. Computer environments typically require the selection of a widget from a predefined set and then the placing and sizing of the widget on the screen. Second widget selection forces the designer to make decisions about the specific nature of a feature too early, as computer environments do not provide abstract placeholders for elements to be described ambiguously (Landay, 1996). Informality and ambiguity are important during preliminary design. For example at the early stages of an interface design it may be sufficient to draw a box named 'contact information'. This will later be detailed as name, address lines, phone email etc, but at the initial stages it is not important enough to waste time on the detail.

3 Computer Supported Sketching

True computer supported sketching is not common. We suggest that the reasons for this are two-fold. To be useful a computer environment must add something to the process (Gross, 1998) and the interface must not add cognitive overhead to the design generation process.

Computers can clearly contribute to sketching by overcoming the disadvantages of paper and whiteboards. A computer can provide limitless space and although the viewing space is limited, zooming can provide an overview. Easy editing can also be provided by software, and the digital artefact is simple to store and transfer.

The other contributions that computer interfaces can make to the process are: to provide intelligent support for transforming the sketch to the formal diagrams that are used as the design process proceeds, for fields where the design describes a process, to emulate some of the functionality of the design while it is in sketch form. The challenge is for the computer to be able to accurately recognise the sketch icons and then transform and animate them.

Providing an interaction device that does not interfere with the sketching process is the other major challenge for computer supported sketching. We start with the premise that sketching must be fast, direct and natural. In the 1960s some work was done with providing direct manipulation interfaces using light pens on cathode-ray tubes. As these screens were phased out in the 1970s and the mouse became the dominant pointing device, the research effort slowed (Gross, 1998). More recently there has been a revival with the development of pen based computing and large interactive displays.

A number of projects have investigated computer supported sketching in a variety of domains. Landay and others (Landay, 1996, Landay and Myers, 1995, Lin, et al., 2000) developed a user interface and a web page sketch design tool; Gross and Do (1996) created an architects' sketch tool; Stahovich (1998) worked with engineering drawings; and Damm et. al. (2000) developed a CASE tool interface. Each project has taken a different approach but all have provided a pen interface and recognised glyphs for transformation into a formal design or to emulate functionality.

4 Computer Sketch Interface Requirements

From a usability perspective a digital sketch environment should imitate customary tools in most respects. It must facilitate informal, direct and rapid drawing in an uncluttered space, without interruption. Also to be worthwhile the digital surface must provide more functionality than paper or a whiteboard. Below we describe the requirements for a digital sketch tool.

4.1 Usability

First the physical interface must be pen based and direct. Pens are the natural drawing tools for humans. When contact is not direct with the surface, such as is experienced with drawing tablets, there is an element of discontinuity and indirectness that makes for a less than ideal interface.

The visible drawing area needs to be large enough for the intended users to work with, either as individuals or groups and accommodate the detail of the design. To allow the users to concentrate on the design effort, tools and support should be unobtrusive. We suggest a single pen and eraser. Depending on the domain it may be important to be able to select ink colour or brush properties, but these should be added only if designers in that field commonly use them.

Clearly, the ability to edit sketches is one of the advantages of a digital environment so support for copying and resizing would be expected. The objective must be to integrate these facilities into a pen interface in a natural way so that little cognitive overhead or training is required.

In the following section we will discuss intelligent recognition, however we contend that while constructing a sketch the users should not be distracted with checking whether the recognition engine has correctly interpreted their pen strokes. This is contrary to the normal view on user feedback. The best recognition techniques available do not guarantee accurate recognition, and what is more the designer may draw intermediary ideas that they do not intend to be carried forward to the next stage. We believe that giving the user feedback on whether a glyph has been recognised will distract them from the design process and runs contrary to providing a rapid, uninterrupted drawing environment.

The obvious difference between pen based computing and standard desktop computers is the replacement of the mouse and keyboard by a pen. Pens function in a similar way to a mouse, except that movement is tracked only when the pen is in contact with the interface; this is roughly equivalent to a mouse down, mouse move, mouse up event sequence. On standard computers keyboards facilitate text entry. While it is possible to have both a keyboard and pen, moving focus from the pen to the keyboard interrupts the design session. This can be avoided by integrating character recognition into software so that the pen can be used to input text.

4.2 Functionality

In order for a computer-supported environment to be worthwhile it must add something to the process it is supporting. Many of the features we are familiar with in standard applications, such as the ability to save, retrieve and edit are likely to be useful.

The other substantive way in which computers can enhance the design process is by integrating sketching into standard applications and by adding functionality to sketch. In order to do this the software must recognise the glyphs. Sketch recognition is a non-trivial task. One of the strengths of sketching is that precision is not required; this is clearly counter-productive when attempting to interpret a sketch. However, once recognised a sketch can be transformed into a formal diagram or demonstrate functionality. For example Knight (Damm et al, 2000) converts sketches to a CASE tool diagram and Denim (Lin, et al., 2000), a web page design tool allows designers to link buttons to web pages so that the designers can ‘walk through’ the page structure while in the sketch environment.

5 Direct pen input devices

There is a range of direct pen input devices currently available; the major discriminating factor is size. PDAs are perhaps the most popular, however for design purposes their screens are too small. Tablet PCs that have a screen about the size of an A4 page would be suitable for single user sketch environments, but are clearly not big enough for a group space.

Xerox developed the ‘Liveboard’ interactive whiteboard in 1990, which they subsequently used as a base for meeting support software (Pedersen, et al., 1993). Smartboard (2000) is a commercially available large interactive display. To support this work we have constructed a low cost large interactive display screen (LIDS)(Apperley, et al., 2001). It is comprised of a standard data projector, rear projected onto a screen approximately 900mm wide by 1200mm high with a Mimio (1999) whiteboard digitiser attached to the screen to provide the interaction. The Mimio pens are used in mouse emulation mode.

6 A sketch interface for a programming environment

GUI interfaces are the standard for most commercial software and their design is a critical factor in the usability of the software. With the exception of very large organisations these interfaces are usually designed and created by software engineers, many of whom have no formal training in interface design. While they may accept that there are some advantages in sketching designs, most of them are reluctant to do so because they see it as a waste of time. This is also true of student programmers. We believe that by integrating a sketching environment that will electronically transpose a sketch into a GUI form, into a programming environment software engineers are more likely to sketch a design and in doing so consider user interface requirements before they create the form.

We are developing a sketch interface for Visual Basic™ (VB) to test this hypothesis using the LIDS described above as the interface. The software has three major components; a sketch space, recognition engine, and a form creator.

The sketch space is a deliberately minimalist environment where the users can draw, handwrite and edit. In drawing mode the user can sketch freely but should ultimately end up with glyphs that roughly depict the VB controls that they wish to represent. In handwriting mode the user pens text that is interpreted by a handwriting recognition module. In edit mode the user can cut, copy, paste or resize sketch components. Our goal is to provide a comprehensive and intuitive design environment where all the interaction is via the pen and LIDS.

Rubine’s (1991) pattern recognition algorithm has been implemented for glyph and gesture recognition. The sketch space operates in the three separate modes to improve recognition, with intelligent mode switching whenever possible. Although this algorithm does successfully recognise graffiti script (Palm Computing, 1994) the intention is to integrate an available handwriting recognition module.

Pen strokes are recognised immediately after completion. Most sketch systems provide some form of feedback, tidying the sketch, changing the colour or displaying the name of the recognised glyph. As suggested earlier, we are experimenting with delaying this until the user

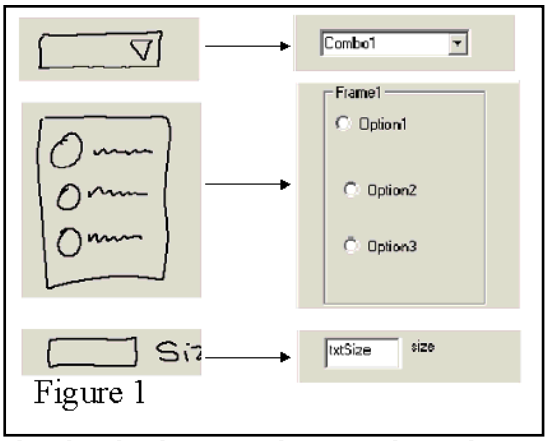

Figure 1

decides the sketch is complete. In edit mode gestures invoke an immediate response with the sketch being redrawn to reflect the gesture request.

On completion, the user instructs the software to create a VB form from the sketch. At this point a rule-based system is used to work out the relationships between glyphs. There are essentially three categories of relationships. Firstly when two glyphs combine to create one VB control, for example a long rectangle with a triangle inside will be mapped to a drop down combo. Secondly, container controls, for example frames, are large rectangles containing other controls. This is particularly important for option buttons where each contained set is mutually exclusive. Finally, VB controls such as text boxes do not have a caption property, so we intend to link labels beside or above such controls to intelligently name the control.

Before the form is created the user has the ability to alter the recognition engine's decisions, changing the type of control, relationships or text. Once this is completed a VB form is generated. A fuller explanation can be found elsewhere (Plimmer and Apperley, 2001).

We are actively developing the environment described above; at the time of writing the sketch space is operational with drawing and editing. Rubine's algorithm has been implemented along with an interface to create and edit gesture sets. Integration into VB has been implemented.

The next steps are: to usability test the sketching environment, build the rule system to combine glyphs and allow users to define the mapping to VB controls. We also plan to integrate an appropriate handwriting recognition module.

7 Conclusions

Sketching is a valuable part of the design process. It allows the designer to quickly represent their design ideas in a physical medium. Although most designs are rendered on a computer, most designers choose not to use a computer for the first stage of design because the currently available interfaces do not support the informality of sketching. We have described the essential elements of a sketch. We contend that unlike most interfaces a sketch environment should delay feedback until the user requests it. We are developing a sketch interface for Visual Basic to test these ideas.

8 References

APPERLEY, M., DAHLBERG, B., JEFFERIES, A., PAINE, L., PHILLIPS, M. and ROGERS, B., (2001): Lightweight capture of presentations for review, *To appear in Proc.* IHM-HCI, Lille, ACM

BEKKER, M. M., (1993): Representational issues related to communication in design teams, *Proc.* Chi '93, Interact '93 and Chi '93 Conference companion on human factors in computing systems, 151-152, ACM

DAMM, C. H., HANSEN, K. M. and THOMSEN, M., (2000): Tools support for cooperative object-oriented design: Gesture based modeling on and electronic whiteboard, *Proc.* Chi 2000, 518-525, ACM

GOEL, V., (1995): *Sketches of thought*, Cambridge, Massachusetts, The MIT Press

GOLDSCHMIDT, G., (1999): The backtalk of self-generated sketches, In *Visual and spatial reasoning in design*, 163-184, GERO, J. S. and TVERSKY, B. (eds), Sydney, Key Centre, University of Sydney

GROSS, M., (1998): The proverbial back of an envelope, *IEEE Intelligent Systems*, 13 (3) 10-13

GROSS, M. and DO, E. Y.-L., (1996): Ambiguous intentions: a paper-like interface for creative design, *Proc.* UIST '96, Seattle Washington, 183-192, ACM

LANDAY, J., (1996): Interactive sketching for the early design stages of user interface design,PhD., Carnegie Mellon University, Pittsburgh, PA

LANDAY, J. and MYERS, B., (1995): Interactive sketching for the early stages of user interface design, *Proc.* Chi '95 Mosaic of Creativity, ACM, 43-50,

LIN, J., NEWMAN, M. W., HONG, J. I. and LANDAY, J. A., (2000): Denim: Finding a tighter fit between tools and practice for web design, *Proc.* Chi 2000, 510-517, ACM

MIMIO, (1999): *Mimio*, http://www.virtualink.com

PALM_COMPUTING, (1994): http://www.palm.com

PEDERSEN, E. R., MCCALL, K., MORAN, T. P. and HALASZ, F. G., (1993): Tivoli: An electronic whiteboard for informal workgroup meetings, *Proc.* Interchi '93, 391-398, ACM

PLIMMER, B. E. and APPERLEY, M., (2001): From sketch to form on a large interactive display surface, *Proc.* National Advisory Committee on Computing Qualifications, Napier, 371-374,

RUBINE, D., (1991): Specifying gestures by example, *Proc.* Proceedings of Siggraph '91, 329-337, ACM

SMARTBOARD, (2000): *SmartBoard*, http://www.smarttech.com

STAHOVICH, T. F., (1998): The engineering sketch, *IEEE Intelligent Systems*, 13 (3) 17-19

TVERSKY, B., (1999): What does drawing reveal about thinking, *Proc.* Visual and spatial reasoning in design, Cambridge USA,