Drawing on the table: computer aided design for dummies

Duncan Cavens
University of British Columbia
Interdisciplinary Studies
Vancouver, BC Canada
604 822 4148
dcavens@interchange.ubc.ca

ABSTRACT
This paper presents a prototype for an interface device which leverages the strengths of commercially available computer-aided-design (CAD) packages and the skills of pen-based designers in a single system. Using relatively conventional technology, a cathode ray tube (CRT) display and a touchscreen, the device connects the traditional skills of a designer with the power and advantages of a CAD system. In order to closely emulate a standard drawing surface, the system interprets pen-based sketching as vector linework. In addition, a user interface system was developed that adapts the current graphical user interface (GUI) paradigm to the particular demands of a large drafting surface, using a two handed interface. While the author only created a smaller proof-of-concept device, the results were positive, and point towards the need for further development.

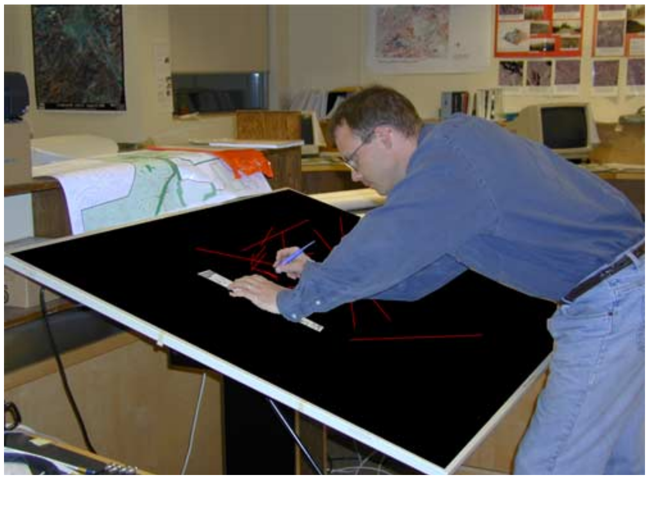
Keywords
Computer-aided-design, pen sketching, drafting, two handed interface

1 INTRODUCTION AND CONTEXT

As with many industries, the introduction of computers into the field of architectural and landscape design has been a mixed success. Most design firms now use computer software for a majority of their day-to-day drafting and graphical work. Computer aided design (CAD) software is used at many stages of a design project: drafting, revisions, sharing of designs between trades, and for production of construction details and presentation drawings. While at first glance it appears that CAD has permeated all aspects of landscape and architectural practise, it is interesting to note that the use of the computer is largely excluded from the fundamental aspect of this industry: creative design. Based upon anecdotal evidence, pencil and paper are the dominant media for initial design work in most North American firms (particularly in the field of landscape architecture.) Once a design has been finalized using 'traditional' methods, it is handed over to a junior staff member who enters it into a CAD system. In essence, the computer aided design system has become a computer aided drafting system.

There are many possible explanations for this use of CAD software. Often, it is the senior associates or partners who are responsible for design. As these practitioners are generally older, they are less likely to have the time or inclination to learn the intricacies of a CAD system, a process which generally takes several years[1]. As well, even with trained staff, studies have shown that overall productivity only increases by a maximum of 5% with a CAD system over a purely manual system[7]. Another important factor is that CAD systems were generally designed for engineering purposes, in particular solid modelling, and adapted to other fields. Many of the tools available do not reflect standard practice in the landscape and architectural design fields. This is also reflected in the lack of usability studies which focus on these fields, when compared to the engineering field (in particular mechanical.)

An obvious question is whether one should care about the



current state of affairs. If users are as efficient using manual methods as computer-aided ones, why shouldn't they continue using pen and paper? While one could argue that true efficiencies have not been gained due to a lackluster interface, there is another fundamental issue. Design is a process of iterative synthesis- the constant testing of ideas against new and different information. Over the course of a large complex projects new information is introduced at many different steps. With the current state of practice, it is difficult to creatively assimilate new data and change one's design to reflect it, due to the disconnect between design and drafting, and due to the large investment in drafting time.

It has been suggested that integrating a designer's pen based skills and knowledge with the power of a CAD system would have powerful synergy[7]. A current theme in computer interface design is the creation of unique interfaces which are designed for a particular purpose, rather than trying to fit the task around the existing interface[5]. This paper presents a prototype for an interface device which leverages the strengths of commercially available CAD packages and the skills of pen-based designers in a single system. It seeks to adapt two components of the traditional drafting system, the drafting table and pen, and connect them to the CAD system and database. This is done using conventional technology: a pressure-sensitive touch screen and a CRT display/desk. In order to closely emulate a standard drawing surface, the system interprets pen-based sketching as vector linework. In addition, a user interface system was developed that adapts the current graphical user interface (GUI) paradigm to the particular demands of a large drafting surface, using a two handed interface.

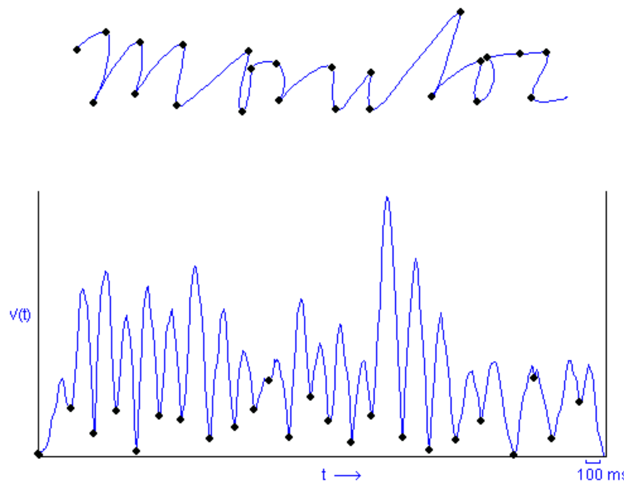


Figure 2: Diagram showing relationships between control points and velocity in handwriting[6].

2 RELATED WORK

There has been similar published work in three closely related areas: examinations of the use and limitations of existing CAD systems; interpretation of pen-based sketching; and in the development of two-handed interfaces.

While Hwang and Ullman [7] focused on mechanical engineers, their detailed analysis of mechanical engineers at the design stage confirms the authors supposition that sketching plays a key role in the design process. The study concluded that, to be effective, a CAD system needed to accepted pen-based input and recognize features. Others, such as Bhavani and John [1] have examined CAD usage (again amongst mechanical engineers) in order to understand the reasons for the lack of productivity in CAD systems. This study in particular concludes that current systems do not reflect the aptitudes and desires of their users, and insinuates that users either need to be completely retrained for existing software, or need a new paradigm for drafting.

Eggli, Bruederlin and Elber present a closely related system in *Sketching as a Solid Modeling Tool*[4]. Their system uses a sketching interface to create three dimensional objects. It interprets the users pen strokes into lines, circles and arcs and then assembles them into three dimensional objects. The paper presents many good points about user interface design issues with regards to pen-based sketching input, as well as good summary of other research in interpretation algorithms for converting points to lines.

There has also been a large volume of research into handwriting recognition, which has similar issues, albeit more complicated, with respect to feature extraction from points. In particular, the work of L.R.B Schomaker [6,9] was the inspiration for the feature extraction algorithm used in the system described here. His work in analysing handwriting is partially based on the realisation that control points in handwriting can be identified by their relative velocity. As demonstrated in figure 1, the writers tend to slow down their pen velocity as they reach a critical point, and accelerate away from the point once they have reached it.

There has also been a fair amount of research into the use of two handed interfaces. Beginning with Buxton and Myers[2], many have demonstrated that users can easily learn and use a two handed interface, and that significant speed increases can be gained by the use of two hands. In addition to providing a good overview of current research in two handed interfaces, Hinckley *et al.* [*] seems to indicate that the non-dominant hand is very good at performing supporting tasks simultaneously, rather than co-dominant tasks.

3 SYSTEM DESIGN

At its fundamental level, the system is a drafting table which uses a back projected screen and is touch sensitive. This allows the user to directly draw on the surface, and immediately sees her drawing reflected on the screen. While the author has only created a smaller prototype of the proposed final system, the system was conceived to be implemented on an industry-standard sized (3' x 4') drawing surface. This allows the user to easily change scales, either by moving one's body position, or via a CAD zoom operation. It also allows the user to design at a relatively small scale over a large area, without constantly needing to pan across a CAD drawing space.

On such a large surface, a rigidly traditional "WIMP" (Windows, Icons, Menus and Pointer) interface begins to break down with respect to usability. As the user will be drawing across the entire surface, having toolbars, menus and dialogs in a fixed location (normally at the top) would require that the user reach and/or move frequently to interact with the system to change modes. Not only is this cumbersome physically, it also serves as a distraction from the task at hand.

As Chatty[3] observes, even with traditional computer aided drawing systems, having to use the same hand for both spatial indication as well as to communicate non-spatial information to the system (as is done in most vector based drawing programs, such as Adobe Illustrator or CorelDRAW) is both inefficient and frustrating. Most current CAD systems attempt to circumvent this inefficiency by the use of keyboard commands. While this is a very successful solution for expert users, keyboard shortcuts take a long time to learn and are not generally used by casual users. In addition, it is difficult to envisage how one could adapt a keyboard interface to a large drawing surface: the user invariably has to stretch and/or move to enter commands, or move the keyboard around to follow her dominant hand, which would be cumbersome.

In order to allow the user to focus on their area of interest, and still be able to switch modes and options, it was decided to implement a floating toolbar that the user is able to summon to their point of interest. The user uses a pen in their dominant hand to draw and input co-ordinate information, and uses their non-dominant hand to summon the a floating toolbar to the hand's location using a double-tap. This division between dominant and non-dominant hand is not rigidly enforced- a tapping motion with the pen will also summon the toolbar. Each button on the toolbar is sized at about 1.5cm x 1.5cm, considerably larger than buttons in a conventional GUI, in order to accommodate the much larger pointing surface of an index finger.

The toolbar 'borrows' its basic functionality and icons from the existing WIMP paradigm and modifies it in slightly different way. Not only was this relatively simple to implement, it also leverages users existing understanding of standard user interfaces to enable them to quickly learn the interface.

An important design element that required careful

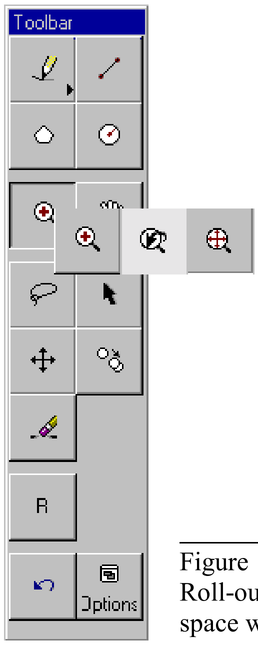


Figure 3: Floating Toolbar design. Roll-out button menus are used to save space while adding functionality.

consideration was how one could interpret the data from the pen-based input into useful information for the CAD system. The strength of a CAD system is its vector-based nature- capturing graphic data as lines, arcs and polygons allows rapid revision and dimensioning. Rather than using the traditional CAD and vector-drawing tool practice of requiring the user to explicitly enter control points, it was decided that maintaining the fluid sketching of pencil and paper was essential. As a result, the system is required to interpret the point data into lines, arcs and circles.

Unlike Eggli's system[4], which automatically interprets and classifies input as line, spline, or circle, it was decided that the user would be required to specify which kind of primitive to draw. This was decided for two closely related reasons:

1) that it would be technically very difficult to implement a system which recognizes different primitives at a high level of reliability.

2) 2) it was reasoned that it would be better not to automatically classify pen strokes than to do it without a very high degree of accuracy.

There was a desire to avoid the frustrations that users experience when the computer incorrectly interprets the user's intentions on a consistent basis (i.e. auto-correct feature in Microsoft Word.) Therefore, a series of buttons were placed on the floating toolbar which allow the user to select which kind of primitive to draw (line, multipart line, closed polygon, arc and circle.) It is felt that the ease with which the user can change types, due to the placement of the toolbar, is a better solution than a partially successful algorithm. Upon informal testing, it was decided to extend the toolbar to allow different 'subtypes' for some primitive types. Similar to Eggli's 'modes', each subtype is a simply a modification of the tolerances within the recognition algorithm to correspond roughly to the following three types:

- *rigid line mode*: the system has a low tolerance for selecting control points, and selects very few
- *freehand mode*: the system selects a large number of control points
- *'inbetween' mode*: the system is moderately tolerant and selects a moderate amount of control points.

All of the settings and tolerances are changeable by the user in a separate pop up window in order to customize the system for individual user's habits and desires.

In addition to the basic drawing commands, it was necessary to implement some basic object manipulation commands such as zoom, pan, select object, move object, copy object and delete object. Each command uses the pen interface to perform spatial selection and indication. The commands use the pen metaphor where possible so as to be more intuitive: for example, when zooming in, the user selects the zoom mode and then quickly circles the area of interest to zoom in on.

4 IMPLEMENTATION

As it was impossible to get access to a large enough touchscreen, the project was implemented as a prototype on a 17" touch screen. A Magictouch pressure sensitive touchscreen was used, which affixes itself to the monitor in front of the screen. As the touchscreen has a maximum resolution of 1024 x 1024, the program was displayed at 1024 x 768 resolution.

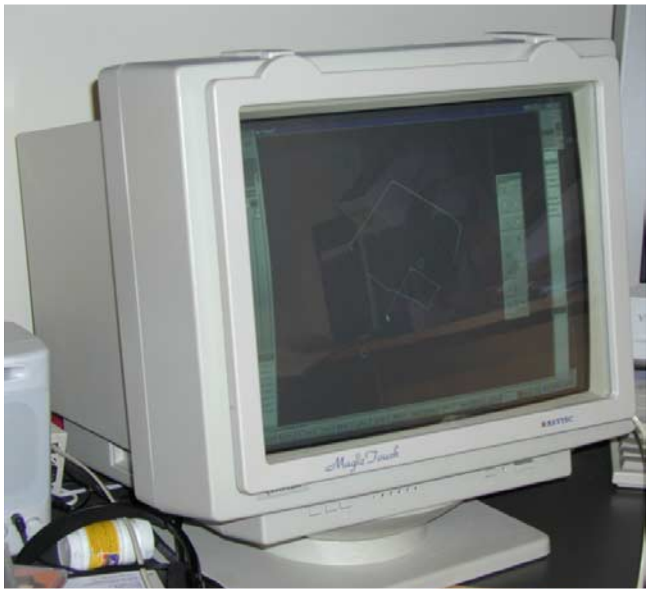


Figure 4: Photo of Mockup System

The touchscreen was attached to a Intel Pentium III system running the Microsoft Windows NT 4.0 Workstation operating system. The drivers supplied with the touchscreen allowed touch input to be interpreted as mouse events. While this made rapid prototyping easier, it also had the effect of precluding what the author believes would be one of the most interesting aspects to touchscreen drafting: the ability to use mechanical aides, such as rulers and templates on top of the screen. These aides help to increase the productivity of mechanical drafters significantly, and it is thought that they would have the same effect on drafters using the touchscreen system. As the touchscreen drivers assume that the user will only touch a single point at a time, placing a ruler on the screen causes the driver to misinterpret the location of the pen.

From the beginning, it was intended that the system should 'piggy-back' on existing CAD software. Modern software design techniques have increasingly allowed software developers to extend and change the interface of the major CAD systems. By using an existing system, it was possible to concentrate development efforts on the user interface, rather than on the drawing database, graphics engine, etc. User's current work could also be brought into the system so that testing could occur on user's current projects, rather than on hypothetical situations. Autodesk, Inc.'s AutoCAD Release 14 was chosen as a base platform as it allows two different kinds of customization: an object oriented access to its database and commands using Microsoft's ActiveX API, and access to the AutoCAD's graphics engine using the scripting language AutoLISP.

Microsoft's Visual Basic language was used to interface with the ActiveX components of AutoCAD. Unfortunately, the customisation experience was not as successful as it should have been. It was only discovered quite a way into the process that the two customisation environments have radically different capabilities, and that it is pratically impossible to communicate between the two environments. As a result, quite a few inelegant techniques at interprocess communication were required, which largely involved saving information to text files in one environment and then loaded back into the same program in the other environment.

Most of the implementation of the user interface was straightforward, although the choice of Visual Basic as a user interface design was perhaps unfortunate. The development environments encourages the use of Microsoft's particular interpretation of the WIMP interface, making it difficult to implement some user interface elements found in other commercial drawing packages (i.e. roll-out button menus.)

In order to extract vector based data from the pen input, an algorithm was written that processes each pen stroke immediately after the user has lifted the pen from the touch surface. Real-time conversion of the input was considered and partially implemented, but was discarded when it was found to be too distracting to the user (i.e. one's attention was focused on how the computer was interpreting the line, as opposed to drawing the line itself.) As the algorithm is based on relative velocity, the unpredictability caused by excessive user concentration causes the algorithm to lose some of its accuracy.

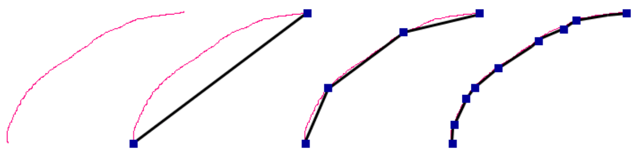


Figure 5: Example of Control Point interpretation. Left to Right: Original, Line, Inbetween, Freehand

As stated above, the feature recognition algorithm is based on the concepts presented by Schomaker[9]. The algorithm tracks the velocity and position of the pen tip during drawing. During the post-processing phase, the velocity is compared against a threshold velocity. If it is lower than this threshold, the algorithm also compares the relative velocity against another threshold to determine if the user is simply drawing very slowly. If the relative threshold is below the threshold, the point is accepted as a control point. A second processing feature was later added to add a minimum distance between control points. It was found that user's tended to pause slightly at key points, such as corners. Their hands were not at complete rest, however, which resulted in a few points being sampled within a few pixels of each other. It is these three parameters which are used to create the sub-types. They are also customisable via an interactive dialog.

Frankly, it was surprising how well the algorithm worked for extracting points from the data stream. Although the algorithm is not as accurate at slow speeds, it works quite well for a first rough cut.

In addition, the software uses the ObjectSnap feature adapted from AutoCAD to connect points if they are within a certain distance (expressed as a percentage of screen area) of a control point. One can select the type of point that one wishes to 'snap' to: end of a line, middle of a line, centre of a circle, or simply the nearest point. These options can be turned off from the options dialog, or the entire snap setup can be turned on from the toolbar.

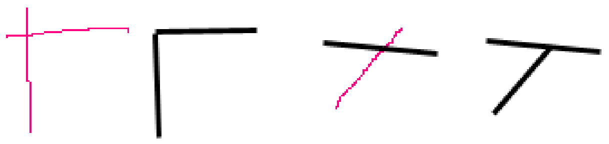


Figure 6: Example of drafting with Object Snap feature turned on.

5 USER TESTING/RESPONSE

A few landscape architecture students were invited to informally test the interface. These students have some exposure to the traditional AutoCAD interface, but do not describe themselves as proficient in the software. Their initial reaction was extremely positive, especially when it was mentioned that the screen was a prototype for a larger system. Use of the toolbar was received generally positively, although one user had problems with the speed required for the double click motion (this is a problem with the touchscreen driver, and was not easily resolved.) In addition, a couple of users wanted to know where certain other commands were (such as splitting a line, rotate, etc..) This raises an important point: with CAD systems contained well over 5000 commands, it is difficult to conceive of how even the most commonly used ones can be accommodated in a single toolbar.

Users were also distracted by the fact that the surface of the touchscreen is about 7mm away from the surface of the monitor used for the test. Depending on the angle of view, it sometimes appears that the location of the pen does not correspond with the point being drawn to the screen. It is anticipated that a different touchscreen technology, where the touchscreen is integrated with the monitor, would fix this problem.

Some users complained that the pause between the time when they lifted the pen from the screen to when the line was redisplayed after being converted to vector format. This time lag, which is noticeable, is a result in a bug in AutoCAD which requires that one regenerate the entire drawing after each operation in order for the changes to be made visible. The processing required to convert the point data into linework does not produce a visible lag.

6 CONCLUSIONS AND FUTURE WORK

While many would regard CAD as a mature technology, it is evident from useability studies that it is not living up to its potential as a design tool. The integration of the CAD technology into traditional drafting tools could go a long way to fixing this situation. While the initial results of this proof-of-concept was promising, there needs to be a few improvements before one can fully test the concept of a virtual drafting table. These areas include:

- Either the touchscreen technology needs to improve, or the drivers for the touchscreen technology need to be found/rewritten to allow the system to accommodate multiple simultaneous contacts with the screen. This would allow users to rest their arms on the screen, use drafting aids, etc. This is especially crucial if one were to migrate to a much larger desk, where the temptation would be to rest oneself on the drawing surface.
- The user interface would need to be designed to accommodate a much larger set of modes and options

in the toolbar. This must be done while still maintaining its' relatively small size, so that all of the options are within easy reach of the hand.

- The algorithm for converting points to lines need to be refined and retested.

REFERENCES

1. Bhavnani, Suresh K. Exploring the unrealized potential of computer-aided drafting. in *Proc. Human Factors in Computing Systems* (Vancouver BC, April 1996), ACM Press, 332-339.

2. Buxton, William. Myers, Brad. A Study in Two-Handed Input in *Proc. CHI* (Boston, MA 1986), ACM Press, 321-326.

3. Chatty, Stephane. Issues and Experience in Designing Two-Handed Interaction *in CHI94-Companion* (Boston, MA 1994), ACM Press, 253-254.

4. Eggli, Lynn et al. Sketching as a Solid Modeling Tool in *Proc. Solid Modeling* (Salt Lake City, UT 1995), ACM Press, 313-321.

5. Hinckley, Ken et al. Two Handed Direct Manipulation. in *Transactions of Computer Human Interaction*, 5,3 (Sept 1998), ACM Press, 260-302.

6. Handwriting Recognition Group, NICI (Nijmegen Institute for Cognition and Information) Web Site. Online at http://hwr.nici.kun.nl/pen-computing/.

7. Hwang, T. Ullman, D. The design capture system: capturing back-of-the-envelope sketches. *Journal for Engineering Design* 1,4, 1990.

8. Majchrzak, A. Effect of CAD on the jobs and drafters and engineers: a quantitative case study. *International Journal of Man-Machine Studie, 32,*3 (1990), 245-62.

9. Schomaker, L.R.B., & Teulings, H.-L. (1990). A Handwriting Recognition System based on the Properties and Architectures of the Human Motor System. *Proceedings of the International Workshop on Frontiers in Handwriting Recognition (IWFHR).* Montreal: CENPARMI Concordia. 195-211.