A novel gesture-based calculator and its design principles

Will Thimbleby & Harold Thimbleby
Department of Computer Science
University of Wales Swansea
will@thimbleby.net harold@thimbleby.net

A novel calculator, designed primarily for interactive whiteboards and pen-based devices, provides a better task fit than conventional approaches. The calculator provides a natural, dynamic method of doing calculations by handwriting using conventional notation. This paper discusses the calculator's underlying design principles, which collectively create a coherent and innovative 'look and feel.' The principle set could be used to help improve user interfaces for other domains.

Calculators, Design principles, Gesture based interfaces, Equal opportunity, Whiteboard interaction.

1. INTRODUCTION

There is evidence that handheld calculators are difficult to use and are fundamentally non-mathematical [2, 5]. A simple example is that operators often have to be transposed by the user (consider a sum like calculating 4×–5, which has to be entered as 4×5±= on most calculators). Although user interfaces for calculators are constrained by ergonomics (screen legibility, button size), their implementing technology has no such restrictions. This is particularly true when handheld calculators are simulated on PCs — and the technology has moved on considerably since the 1970s, which was the determining era for handheld calculators.

We have developed a new calculator with improved task fit with mathematics, and thus we have broken out of the traditional design approach. Pleasingly, the calculator is very engaging to use — where as another paper [7] discusses its evaluation, this paper presents its new design principles. The success of the new design suggests that the principles, in their particular combinations, might be usefully applied in other domains.

1.1 Overview of the interface

Imagine writing a calculation down on paper, and the paper magically working out the answers. Our calculator works like this, using an approach that is ideal for gesture-based user interfaces, for handhelds with pens to interactive whiteboard use in classrooms. The calculator is written in pure Java and runs under Windows, Linux and MacOSX, and it works with standard hardware such as Mimio, SMARTboard, or Wacom tablets — it is somewhat tedious to use it with a trackpad or mouse, as it uses normal handwriting as its mode of input. It can be downloaded from http://www.cs.swansea.ac.uk/calculators/, where a movie of it in use is also available.

A different approach can be seen in [8], which includes some useful background this short paper does not have space for. No calculator known to us is as versatile and interactive as ours; indeed, our calculator was selected as an exhibit at the UK Royal Society Summer Science Exhibition in July 2005, and we hope at the conference to report on further insights from evaluation based on its exposure to 4000+ users.

1.2 Evaluation

Without exception everyone (100s of people) who has used the new calculator has liked using it, and many users have found it easier to use than their own handheld calculators. Users find it fun and enjoy using it. Users with little mathematical skill enjoy using our new calculator, and some have grinned when getting it to change, for example, 2^3=8 to 3^2=9. Sophisticated mathematical users have also enjoyed exploring issues such as why 6!+9! is divisible by 100, or pushing the calculator's arithmetic (e.g., finding π from $e^{i\times?}$=–1).

It is important to look at why the design forms such a successful user interface. The principles presented in this paper summarise our principle-led design of the calculator, and are informed by the user testing and evaluation of the prototype. We have reported elsewhere on an empirical evaluation [7].

2. HOW IT WORKS

It is very hard to capture the look and feel of an interactive program, especially an innovative one, in a static medium like paper. This section therefore merely gives a hint of the calculator's capabilities. Figures 1.1–1.6 show a sequence of screen snapshots of it in use. They first show a user doing the sum 3×4; in the first screen shot, the user has written 3×4 and the calculator is "catching up" with their handwriting and has just rendered the 3 in a

typographically neat font. User input is handwritten blue and 'dries' black, thus it is never confused with what is already on the screen as it is written.

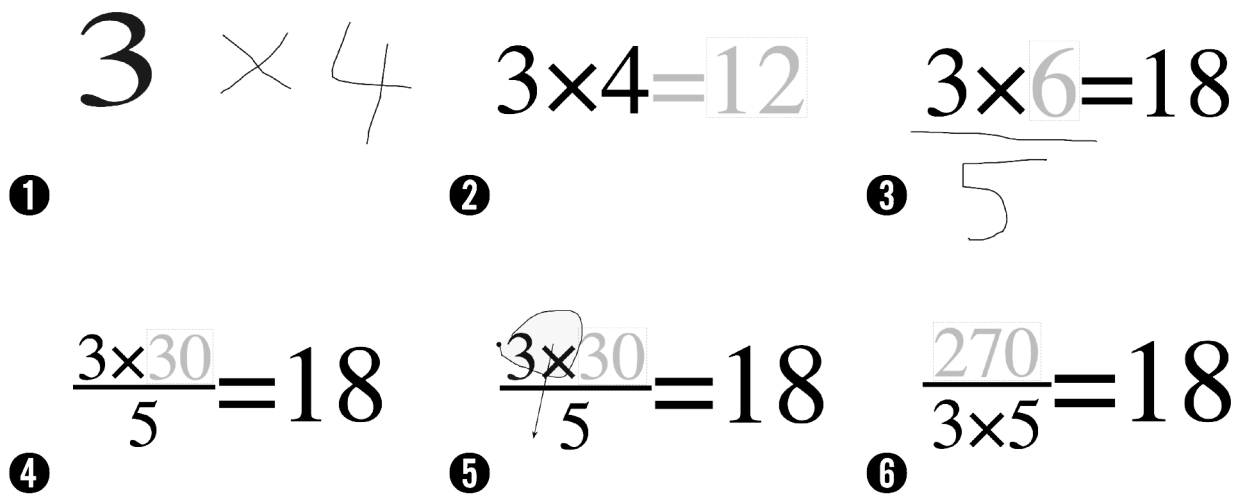

FIGURE 1: A sequence of six consecutive screen shots of the calculator solving various equations. The thin 'sketchy' text (e.g., see Figure 1.1) was written by hand, and as the calculator recognises the handwriting, it is morphed into typeset mathematics (compare Figure 1.1 and 1.2).

FIGURE 2: Using the calculator on a back-projection SMARTboard (6' diagonal) that permits writing using the tip of a finger.

In Figure 1.2, next, the calculator has morphed all the user's input, and immediately combined it with the output (here, '=12') and displayed it all as a typeset equation. The output generated from the calculator is shown in grey in this paper (though typically it is a colour like red in real use). The user continues to edit the equation and by the time of Figure 1.3, they have deleted the 4 and written '=18'. Effectively this poses the question "three times *what* is eighteen?" making the calculator compute "3×?=18". Additionally in Figure 1.3, we can also see the user continuing to edit this solved equation as if it were their own input; they are starting to divide the left hand side by 5.

By Figure 1.4 the calculator has morphed these changes and the combined the typeset output and the user's input into another neatly typeset equation, now showing a generated 30. Had a user wished to perform this calculation on a conventional calculator, they would have had to have entered it in a particular order and with a final = sign, such as 18×5/3=.

In Figures 1.5 and 1.6, the user "drag selects" the "3×" from the previous screen and drags it below the division line. (This is an "ink" edit, the "3×" is not a syntactically nor semantically meaningful unit — see below.) Finally, Figure 1.6 shows the result of this edit, and it is mathematically instantly correct — thus providing a solution to ?/(3×5)=18. What has been done in one gestural operation on the new calculator would have needed around 14 keystrokes on a conventional calculator that permitted last-calculation editing (e.g., so-called twin line display calculators); moreover, at every step except the last, on a conventional calculator, the expression would be wrong, whereas on the new calculator *every* intermediate step is a valid calculation.

The calculator has other features, which are not the concern of this paper; for example, there is a wastebasket to delete anything by conventional drag & drop; a dock (visible in Figure 2 on the left) can be used for selecting, storing equations and values; there are some features used for teaching purposes; and there is an 'analogue clock' that is used for undo.

3. PRINCIPLES

The basic style of interaction is called *equal opportunity*, and it certainly lends itself to arithmetic calculations [5] and other applications [3]. What is new here is the effective combination of two dimensional, WYSIWYG, gesture based, instant behaviour, and morphing, that *taken collectively* make a coherent set of features that combine extremely well for the task domain (and perhaps for other domains, unfortunately beyond the scope of a short paper). But, further, the design introduces new design principles: *ink editing* and (non-trivial) *instantly declarative interfaces*.

3.1 Standard HCI principles

The calculator was designed bearing in mind a range of important but conventional HCI principles: it should fit the task domain (and all that that implies) and reduce the user's short-term memory workload; etc. Conventional calculators do very much worse in supporting these principles. Additionally, our new calculator design draws on standard concepts such as undo, affordance, modelessness, and avoiding error messages (by invariantly ensuring correct partial evaluation).

WYSIWYG usually means that when you print, you get what you have on the screen. For interaction, what you see is what you *have* got is more important [6] (i.e., WYSIWYH), a variation that is a principle for interaction, not for quality display or printing, for example. For calculators the interaction problem is worse, if something is calculated based on a hidden preference of an implicit operator, will the user ever realise?

Our calculator always uses explicit operators where there could be a misunderstanding over the implicit annotation or operator, for example, it inserts an explicit multiplication between a ")" and a "(". Rather than leave the user with their hand-written input, the calculator converts everything to a typeset, well laid-out mathematical expression, and this allows the users to know with certainty what is being computed, instead of wondering whether they have entered it correctly, whether the computer is recognising their handwriting correctly, or whether there is some invisible mode or data affecting the result.

In short, there is no hidden information or state, and all visible information is used. The calculator shows *exactly* what is being computed, thus there is no confusion for the user. Although these and other familiar principles (e.g. undo) have been consciously combined in an unusually coherent way, space precludes a full discussion of their application.

3.2 Ink editing — not syntactic editing

Instead of forcing a user to think syntactically about the structure of how the mathematics works to edit an expression (even if the average user knows what that means!), the new calculator lets a user interact flexibly with the actual ink used. Of course, conventional calculators *very* severely limit what editing is — it is limited to appending characters or deleting the last number or the *entire* calculation.

The new calculator does not restrict what is selected to be adjacent or to have any particular syntactic structure; it is easy, for instance, to select alternate digits out of a number and move them elsewhere in an equation. Although that seems contrived, the ease and naturalness is important: consider editing 31.416 to 3.1416, which can be done directly by moving either the decimal point or the second digit. (In contrast these two operations would, if permitted, be very difficult in a syntactically constrained editor.) On conventional editing calculators, what is a single operation here has to be broken down into a more tedious sequence of operations like delete-move-insert or equivalent.

Although the two-dimensional notation of mathematics implies many syntactic relations, the new calculator does not impose any. Instead, ink editing allows the user to edit their work naturally as a picture, in a way that is impossible with any one-dimensional (conventional) representations. This flexible ink editing of two-dimensional notation allows the user to rearrange and edit mathematics semantically or syntactically, in a way that can be very close to how the user thinks about the abstract mathematics. The advantages of picture editing were forcefully described in [1].

WYSIWYH means that the semantics of any expression is directly linked to the syntactic "ink." For example, drawing a horizontal line might mean either a division sign or a subtraction, depending on what the user means. The calculator disambiguates, by allowing either interpretation, *which can be changed at any time*. If numbers are written above or below a subtraction line, the line becomes a division symbol; and if they are both deleted, it reverts to subtraction.

3.3 Instantly declarative

The essence of an *instantly declarative* interface is that it cannot show the user something that is false, *ever*. For example, an instantly declarative calculator could never show "3+4=15." The display has to be correct without any further user action; and instantly. The benefits for the user are obvious: there is no confusion for the user, the input and output *always* correspond. The interface feels natural and immediately responsive to user input.

This approach means that "=," "Go" (and similar) buttons are redundant; they only slow the user down (sometimes users get stuck, waiting for things that will not happen until they do something which they don't know they are

supposed to do). An instantly declarative interface has to cope with partial and incomplete input and respond fully in a timely fashion. Using *equal opportunity* [3] enables us to handle incomplete input or partially complete equations. Requiring complete input can lead to a very modal user interaction. This may be suitable in some domains (e.g., with safety related issues), but it is in principle unnecessary. Conventional calculator designs never escaped this unnecessary modality of requiring complete input.

3.4 Output = input = everything

A declarative calculator, both from our informal and empirical evaluations, is superior. But the power of a declarative interface is only fully realised when combined with a two-way equivalence between the user's input actions and the system's output. This added to the *equal opportunity* that treats output and input equally, creates a uniquely usable interface. Users are suddenly able to solve problems, such as 'what power of 2 is 56?' (i.e., $2^{x}=56$) *directly* that they might have no idea of how to solve otherwise, and which would be impossible without circumlocution — and would be impossible to do correctly without prior experimentation on a calculator, for even within-brands do advanced arithmetic calculations differently!

The 'output = input' concept works smoothly with a calculator, because the output and input are the same format and can be combined in the same expression. With our calculator, a user can replace the computer output with their own, and nothing will change. This means that if the user writes the correct answer in then the calculator shows no extra work, and it means that if the user writes a wrong sum like "3+4=15" the calculator corrects it. When users use it they find that as one user put it their old calculators are "nagging and pedestrian fusspots."

3.5 Continuous feedback

The visibility of the system's status is provided through two kinds of feedback: *annotation* and *morphing*. These together provide clear feedback about exactly what is happening with the user's input and the calculation. Throughout the calculation the calculator morphs the input into a neatly typeset output equation. Without this linking of the output to the input the user has a jarring experience that leaves them wondering where the output came from. The morphing provides continuity between the user's input and the typeset equation that allows them to continue to edit and use it. Certainly, the animation in the morphing is visually seductive.

4 CONCLUSIONS

Although the new calculator furnishes a very simple user interface to a boring application (who is *really* interested in calculators?) it is very engaging — and this is true despite the calculator's prototype implementation's shortcomings, particularly its imperfect handwriting recognition.

We started with a principle-led design, but ended up with a user interface that is surprisingly effective, one that is fun and engaging to use, and one that uses and develops a new style of interaction. Whether the new style of interaction can be successfully generalised into other domains remains to be seen, but certainly the individual principles that led the design can be used to their benefit.

ACKNOWLEDGMENTS

Will Thimbleby is supported by a Swansea University PhD scholarship. Harold Thimbleby is a Royal Society-Wolfson-Research Merit Award holder, and acknowledges this generous support.

REFERENCES

[1] R Bornat & H Thimbleby, (1992) "The Life and Times of Ded, Display Editor," *Cognitive Ergonomics and Human Computer Interaction*, pp225–255, J. B. Long & A. Whitefield, eds, Cambridge University Press.
[2] P. Cairns, S. Wali & H. Thimbleby, (2004) "Evaluating a Novel Calculator Interface," *Proceedings BCS HCI Conference*, 2, edited by A. Dearden & L. Watts, Research Press International, pp9–12.
[3] C Runciman & H Thimbleby, (1986) "Equal Opportunity Interactive Systems," *International Journal of Man-Machine Studies*, **25**(4):439–451.
[4] H Thimbleby (1996) "A New Calculator and Why it is Necessary," *Computer Journal*, **38**(6):418–433.
[5] H Thimbleby (2000), "Calculators are Needlessly Bad," *International Journal of Human-Computer Studies*, **52**(6):1031–1069.
[6] H Thimbleby (1983) "'What You See is What You Have Got'–A User-Engineering Principle for Manipulative Display?" First German ACM Conference on Software Ergonomics, *Proceedings ACM German Chapter*, **14**:70–84.
[7] W Thimbleby, (2004) "A novel pen-based calculator and its evaluation," *Proceedings Third Nordic Conference on Human-Computer Interaction*, ACM NordiCHI, pp445–448.
[8] J. LaViola, Jr. & R. Zeleznik (2004) "MathPad2: a system for the creation and exploration of mathematical sketches" *ACM Transactions on Graphics*, **23**(3):432–440.