EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2008)
C. Alvarado and M.- P. Cani (Editors)

MathBrush: A Case Study for Pen-based Interactive Mathematics

George Labahn, Edward Lank, Mirette Marzouk, Andrea Bunt, Scott MacLean, and David Tausky

David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, ON, Canada, N2L 3G1
<glabahn, lank, msmarzouk, abunt, smaclean, datausky>@uwaterloo.ca

Abstract

Current generations of computer algebra systems require users to transform two dimensional math expressions into one dimensional strings, to master complex sets of commands, and to analyze lengthy output strings for relevant information. MathBrush is a system, designed based on research in education pedagogy, that provides a pen-based interface to many of the features of computer algebra systems. We describe relevant work in education pedagogy as a motivation for MathBrush's design. We highlight aspects of MathBrush that are unique from other contemporary pen-math systems. Finally, we present the results of a thinkaloud evaluation of the MathBrush system. Together, these observations validate aspects of the current design of MathBrush, suggest areas for refinement, and inform the design of future pen-math systems.

Categories and Subject Descriptors (according to ACM CCS): G.4 [Mathematical Software]: User interfaces

1. Introduction

Tablet computers, electronic whiteboards, and other pen-input computer interfaces present opportunities to alter the way that people use computers in education and work environments. One domain where such a transformation is possible is in mathematics. In contexts as diverse as high-school classrooms, engineering firms, and university research labs, individuals perform complex mathematical problem-solving tasks, frequently using pen and paper. When pen and paper fails to support their problem-solving tasks, these individuals can turn to computer algebra systems (CAS) to solve computationally those problems that are either too tedious or too complex to solve by hand. There exists, however, a mismatch between mathematical work done on pen and paper and mathematical work performed in a CAS. Specifically, the need to transcribe mathematical expressions, an inherently two-dimensional arrangement of symbols on a page, into a one-dimensional sequential form injects awkwardness into the interface [Rut02].

This paper describes the design rationale of our pen-math system, MathBrush. The goal of MathBrush is to support mathematical tasks, common to both researchers and students, which are currently done on computer using a CAS. As a result, we seek both to design an effective system for pen-based access to CAS and to study recognition technology and interface designs that most effectively support pen-math. MathBrush includes a pen-based interface to draw freehand math expressions in a similar fashion as is currently done on pen and paper. Recognition algorithms support the transformation of hand-drawn math into MathML [CIPe01], a mark-up language that represents the underlying hand-drawn math expression. MathBrush also includes a number of features that support the tuning of the interface to specific users (e.g. trainable recognizers), and the study of math recognition algorithms (e.g. a component-based architecture and an interface for regression and acceptance testing). Finally, to allow mathematical tasks common to both researchers and students, MathBrush supports the input and output of large expressions, and process logging to capture and archive problem-solving rationale.

MathBrush is unique from existing pen-math systems [vDO, LZ04, XTh] in a number of ways. First, MathBrush is focused around the existence of back-end CAS. As a result, the entry and recognition front-end of MathBrush in-

cludes little mathematical knowledge: specifically, we implemented no mathematical algorithms. Instead, all mathematical manipulations are performed via menu systems that expose the underlying functionalities of CAS. As well, unlike previous systems that support a single operation on an input, such as numerical evaluation or solving for an individual variable [vDO], MathBrush includes features that permit repeated manipulation of mathematical expressions using natural gestures and context-sensitive menus. The inclusion of a full-featured CAS and the ability to repeatedly manipulate mathematical expressions are justified based on an extensive review of relevant pedagogical research into the use of computers in mathematical education, particularly at the high-school level. Researchers in this area have repeatedly noted that both advanced CAS features (e.g. expansion, substitution, simplification, factoring, derivatives, etc.) and repeated manipulation of expressions (e.g. expand an expression, then simplify it, and then factor it) are necessary for even high-school level algebraic and pre-calculus instruction [Tay95, PS01, Art02, Rut02, PHG04, LPE02].

In this paper, we describe the rationale for the design of the MathBrush system, using education pedagogy as a motivation. We highlight various properties necessary for realistic pen-math systems, and describe how MathBrush supports these features. We also describe the unique features of MathBrush. Finally, we evaluate our design using thinkalouds, and present both the strengths of MathBrush and areas where additional work is needed. Together, the data in the paper provide guidance for on-going work in the design of pen-math systems.

This paper is organized as follows. First, we survey existing systems for pen-math, including systems that support the entry of equations for display and systems that support interactive math. We also examine research on the use of computers in mathematics education. Next, we describe the design of MathBrush. Finally, we describe the preliminary results of an on-going usability study on the MathBrush system.

2. Related Work

2.1. Pen-Math Systems

Current pen-math systems for use on devices such as Tablet PCs can be characterized based on two distinct primary goals. A set of systems exist that seek to simplify the process of creating typeset math expressions on computers. Other systems, including our MathBrush system, seek to recognize math expressions for the purposes of performing mathematical operations through a pen-based interface.

2.1.1. Typesetting Math

Many computer programs, including LaTeX, MS Word, and Internet Explorer, support the display of typeset mathematical expressions. To create typeset expressions, a user typically transforms the desired two-dimensional expression

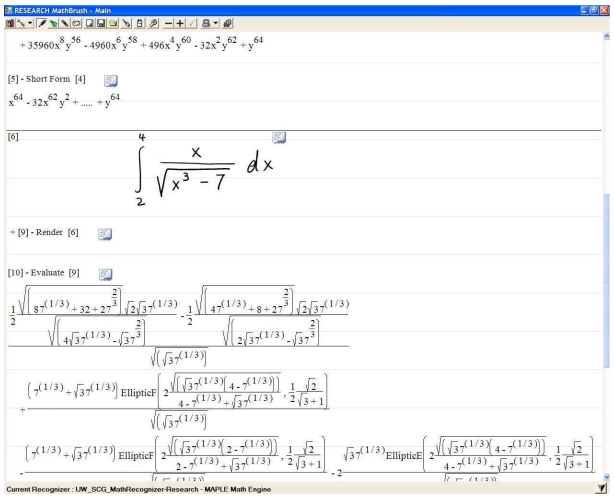

Figure 1: *MathBrush User Interface, showing both input and the rendering of a long output string. The output display heuristics try to keep fractions, parentheses, etc. together, and prohibit horizontal scrolling. For example, for fractions, as line-breaking becomes necessary, the system may transform rendering of a fraction from* $\frac{numerator}{denominator}$ *to* $\frac{1}{denominator} * (brokenNumerator)$*, or to* $(brokenNumerator)/(brokenDenominator)$ *depending on the length of numerator and/or denominator.*

into a one-dimensional text string. Both characters and commands are combined, either through some interface as in Microsoft's Equation Editor or within the text string as in LaTeX, to produce an expression that contains the characters arranged appropriately as specified by the commands.

A set of research systems exist that support the creation of these typeset equations using pen-computers [CY99, SFUK03, Smi99]. These systems were motivated by two factors. First, there is an informal understanding that, because math is taught and typically expressed on pen and paper using its two-dimensional notation, expressions can be more accurately input and more easily expressed by people to a computer for typesetting if they can use the same notation [Smi99]. Second research has been done comparing the speed of diagram entry for math and other diagram notations using text, a mouse and command palette, and a pen [AK93], and the pen-based interface was found to be approximately twice as fast.

2.1.2. Pen-Based Math Manipulation

Computer-based mathematical manipulation is normally performed using CAS such as Maple, Mathematica, NuCalc, or Microsoft Math. Each of these systems support the manipulation of equations of varying complexity. Maple and Mathematica are "full-featured" CAS, providing sophisticated mathematical reasoning tools. Microsoft Math does

not support repeated manipulation of equations; instead, the system's focus is on tutoring users in common mathematical tasks. Finally, NuCalc is an example of an advanced graphing calculator that supports graphing equations and solving equations numerically, graphically, or symbolically.

Some recent research has explored the possibility of using pen-based computers to allow users to develop an understanding of mathematical expressions. The MathPad^2 system [LZ04] is an application for creating mathematical sketches. Users create hand-written mathematical expressions using familiar math notation and draw free-form diagrams. They then create associations between the equations and diagrams. The diagrams are animated, and the mathematical equations specify the behaviour of the animation.

While animating diagrams allows users to see a physical interpretation of abstract mathematical concepts, mathematical manipulations such as evaluating, approximating, expanding, and factoring, represent another area where pen-math systems could aid problem solving. MathJournal [XTh] appears to be the first commercial system for doing mathematics on Tablet PC. MathJournal recognizes and interprets diagrammatic and graphical representations of some engineering and mathematical problems, but has limited mathematical capabilities. Microsoft Math also allows users to enter equations using a *soft input panel* (sip), similar to the sip used for text entry in tablet computers. More recently, the MathReco [vDO] system demonstrated support for evaluating and solving mathematical expressions. The system included a rule-based recognizer for mathematical expressions, and typical editing features. It also allows an input expression to be graphed, a single variable within an equation to be evaluated, or a definite integral to be approximated. These operations were accomplished using either a built-in math engine or Mathematica, a CAS. However, once the math engine has evaluated or approximated the output, no further manipulation of the output is supported. One open question raised by systems such as MathJournal and MathReco is how much mathematical problem-solving power must be incorporated into a pen-math system.

2.2. Computers in Math Education

The goal of the MathBrush system was the design of a full-featured pen-math problem-solving system. We envisage such a system as being useful in education, in engineering, and in research to support a subset of the mathematical problem-solving tasks encountered by individuals in these domains. While each of these domains has unique problem-solving approaches and uses math of differing levels of sophistication, we might assume that high-school math instruction requires lower levels of mathematical sophistication than does engineering or mathematical research. In this section, we present an overview of research on the use of computers in math education, including their benefits, the features needed, and the drawbacks of current technology.

Research on the use of four function, scientific and graphing calculators argues that there are few drawbacks to introducing CAS into educational settings [Tay95]. As well, the widespread adoption of CAS by mathematicians and engineers argues persuasively for the value of these systems to the process of mathematical analysis despite the need to invest time to master the complexity inherent in both input and manipulation during interactive math problem solving [Art02]. In particular, researchers have noted that the use of increasingly sophisticated calculators has particularly benefitted the strongest students, and that the drawbacks for academically weaker students are limited [Tay95].

In their study of 16-year-old students, Leinbach et al. [LPE02] identify the role of a CAS as assuming some of the mechanical processes involved in solving mathematical expressions so that students can spend more time studying conjecture, comparison, and new problem areas. These tasks, which are considered both higher level and desireable in developing advanced mathematical understanding by education researchers [LPE02], require both basic evaluation and advanced CAS features such as expansion, factoring, substitution, and derivatives. Echoing this result, Pierce and Stacy note that the use of a CAS in education and in early research promotes the use of multiple representations, discussion of meaning, and other factors that are considered positive in developing a mathematical understanding of real-world connections to the computations being performed [PS01]. Given this research in education, we can claim that both advanced CAS features and the ability to repeatedly manipulate CAS output are necessary to support high-school math education at, in the case of these studies, the math curriculum for grades ten to twelve.

Given the need for advanced CAS features and repeated manipulations, one might be tempted to claim that a CAS is all that is needed to support mathematics education. However, the interactive features of a CAS are not perfectly suited to the education and research tasks for which a CAS might be used, and significant barriers exist to their adoption in education settings. Pierce et al. [PHG04] identified the need for technical assistance to support students and researchers as they master the use of menus, the location of commands, and the syntax translation necessary for providing input and interacting with CAS. As well, Ruthven et al. [Rut02] noted the difficulty in translating expressions into linear form, a difficulty more often faced by students who struggle with mathematical concepts. Even with pretty-printed verification, there also exists a need to translate the formatted math output between forms that are mathematically equivalent but semiotically distinct, and then to understand what to do with the output given a multitude of CAS features. As a result, having some simplified form of pen-based entry and access to relevant commands would allow verifying correctness of input for computer algebra systems and speed the adoption of these systems by students and researchers [PHG04].

3. Designing MathBrush

The purpose of the MathBrush system was both to design an effective system for pen-based access to CAS and to study recognition technology and interface designs that most effectively support pen-math. In this section, we describe the challenges in building a pen-based system that interacts with the features available in CAS and the decisions we made to satisfy these design challenges. While MathBrush shares many features with existing pen-math systems, MathBrush is the only pen-math system that provides tight integration with a back-end third-party CAS and allows multiple and repeated operations using this back-end CAS. As a result, many of these design decisions are unique to MathBrush.

3.1. Expression Entry and the Display of Output

A computer-based mathematical reasoning system exists primarily to allow users to manipulate an expression or set of expressions that defy easy mechanical solving, not to supplant paper for trivial expressions. As a result, CAS are used to provide support for the entry, interaction, and display of results from expressions that may be large and complex. A pen-math system must support long input, easily achievable through a scrolling canvas. A pen-math system must also support the output of expressions that may be both large and complex. We support this using simple heuristics. Figure 1 shows output of a large expression and describes aspects of the heuristics.

3.2. Accessibility of Interactive Features in CAS

As noted by Leinbach [LPE02], even simple optimization problems and high-school level math can require access to advanced CAS features such as expansion, factoring, substitution, and derivatives. While preserving the pen-based nature of interaction, there exists a need to support access to many different commands. However, mastering the large command set is a significant barrier to adoption of CAS, in both professional [Art02] and educational settings [PHG04].

To limit the number of commands displayed, MathBrush incorporates a context sensitive pop-up menu. The menu is populated by operations based on a cursory analysis of the mathematical expression and the manipulations most likely to be performed. For example, the context menu for matrices, Figure 2, includes operations such as inverse, determinant, and rank, whereas the context menu for polynomial expressions, Figure 3, includes operations such as factor, differentiate, simplify and short form. With the high number of operations available from the CAS, displaying only the operations that are most likely based on analysis of the expression is particularly helpful for less experienced users. More advanced users can extend the context menu to include additional operations as needed.

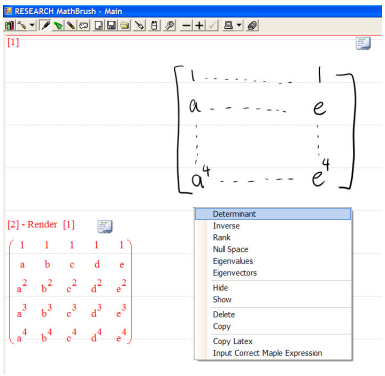

Figure 2: *Context Menu for a Matrix Expression*

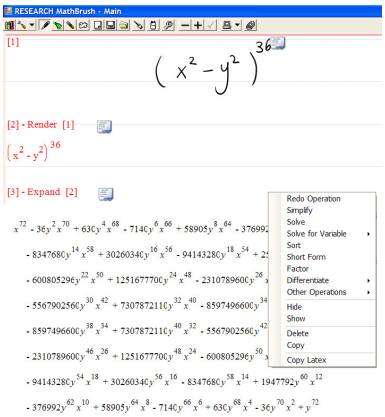

Figure 3: *Context Menu for a Polynomial Expression*

3.3. Interactive Editing in Place of Math Expressions

In CAS, it is common to develop a partial solution, to manipulate some portion of that solution, and then to reintegrate the manipulated subexpression into the pre-existing partial solution. This is necessary even at the high-school level, where a student may wish to substitute a specific instance of a subexpression with a new variable, or to simplify only a subset of the terms in an expression. With CAS, users identify the position of the subexpression(s) they wish to access, then use specific CAS commands to index into the equation to access the subexpression(s). They then issue additional commands to operate separately on the subexpression(s). If the original expression is complex or long, this operation is tedious and error-prone [Rut02]. To allow the user to do such manipulation and editing in MathBrush, we have designed an editing-in-place mechanism that allows interaction with CAS output. Users can circle subexpressions using the pen and use the context menu to display the selected part of the expression in a floating MathBrush window. Subexpression editing is shown in Figure 4.

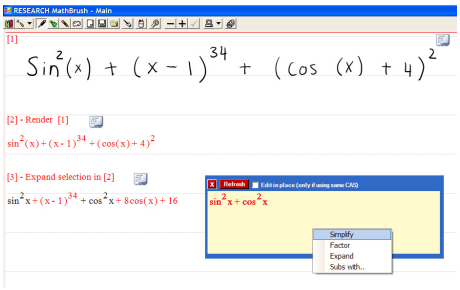

Figure 4: *Output Manipulation in MathBrush*

3.4. Plotting

Creating a graphical representation of an equation (i.e. plotting) is a common feature of many pen-math systems. MathBrush also includes a plotting control that supports two and three dimensional plots, and the pen can be used to rotate these plots to examine them from different angles, a feature particularly useful in 3 dimensional plots. MathBrush augments this control with support for pen-based input to control the range of plotted values on multiple axis. Users can modify the range of any axis by writing a new value near the rendered range value of the axis being modified. Using 2D proximity, we map the range onto the appropriate axis. This functionality is shown in Figure 5.

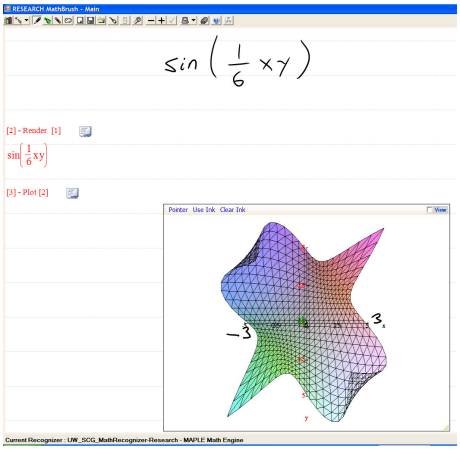

Figure 5: *Plotting in MathBrush. The user is adjusting the x-axis limits from [-5, 5] to [-3, 3], by drawing the new limits on the plot control.*

3.5. Additional Features

There are a series of additional features in MathBrush that were not tested during our thinkaloud study, including the ability to interactively swap CAS and logging. While CAS normally behave identically given the same input, for some inputs different CAS give results that appear quite different (are semiotically distinct), but are mathematically identical. MathBrush deals with this issue by viewing all CAS as pluggable components. A user can select a new CAS to check for alternate answers to their problems. Such a selection can occur anytime during a session. Logging allows users to revisit their past actions. This is useful for mathematical reflection [PS01] and when editing features make it difficult to track exactly how past equations were modified to create a result.

4. Evaluation

4.1. Goal

We were interested in several aspects of our system. First, we wanted to determine how accurately users enter input. Second, we wanted to explore the ability to interpret and correct recognition errors. Finally, in designing our system, we made use of context menus and a subexpression-manipulation panel to enable users to perform an extensive set of mathematical manipulations, including expansion, substitution, simplification, factoring, graphing, and evaluating. While our decisions seemed reasonable during development, initial evaluation can serve to validate these decisions and suggest areas needing refinement.

4.2. Method

We evaluated our system using thinkalouds and a semi-structured interview. Participants were given a five minute introduction to the system. The basic functionality was demonstrated, including the process for drawing equations, validating the recognition results, rendering the recognized results, and performing mathematical operations on the equations. We were careful to show participants how to work with recognition errors in the system, and were honest about the fact that recognition is occasionally brittle.

Participants then entered and manipulated several mathematical equations using a tablet computer (Acer 14.1" screen). These included two algebraic expansions and simplifications, the manipulation of a subexpression within one of the algebraic expansions, the entry of a set of matrices and operations on the matrices, and plotting of a two dimensional trigonometric equation. During these tasks, participants were encouraged to verbalize their thought process. We provided no initial assistance to participants in accomplishing any of their tasks, and intervened only when participants seemed to have exhausted all avenues of exploration to accomplish their task and were becoming frustrated.

Finally, after participants had completed the tasks with the system, a process that took approximately 20 minutes, we conducted a semi-structured interview asking about their current mathematical tasks, and the match between system functionality and their mathematical tasks.

4.3. Participants

Participants for our study were drawn from the undergraduate Faculty of Mathematics at our institution, which includes students studying math, computer science, and various subdisciplines within these fields. The students were undergraduates, and had all taken multiple courses in calculus and linear algebra. As is typical in thinkaloud style descriptive evaluations, we focus on a relatively small sample subject population, and explore users' attitudes in depth [Lew82]. Specifically, we worked with five participants, the number suggested by Nielsen in his writings on early usability evaluation [Nie00, NL93].

5. Observations

We focus our observations around two themes. First, we look at inputting mathematical expressions and validating recognition results. We then present observations of interactions with the mathematical equations to solve specific problems we gave our participants.

5.1. Equation Entry and Recognition

In MathBrush, equation entry is a three step process. Users first draw the mathematical equation. They then interact with the equation in an input validation panel (IVP) to correct any character recognition errors (see Figure 6). Finally, they render the equation to validate structural analysis.

When drawing equations, we observed several notable interactions. First, equation entry was generally not a problem. In all but four instances, participants were able to enter equations successfully. The exceptions were a result of ideosyncracies in individual character recognition, a software programming bug that caused plotting to fail, and spacing issues associated with segmentation of terms in a matrix. One suggestion for correcting character recognition ideosyncracies was to allow use of a soft keyboard to manually enter the correct character.

Some minor issues did appear. One challenge that all users had was the propensity to leave extraneous ink, specifically small dots, when drawing ink equations and then invoking the input validation panel. This extraneous ink could result in errors in parsing that were difficult to understand. We are in the process of refining our parser to include the ability to omit this extraneous ink. As well, two editing features were not used: scratch out and translation. While scratch-out is commonly supported, our participants all used the eraser end of the electronic stylus rather than the scratch-out gesture. As well, the process of circling and moving terms around (i.e., translation), either to allow modification or to correct structural analysis was not used. Participants would, instead, simply choose to erase and then redraw a portion of the equation. These observations, however, were minor, as they did not interfere with participants' overall ability to interact with the system successfully.

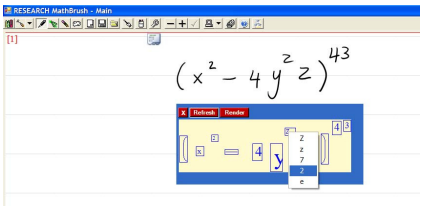

Figure 6: *During recognition, the math equation is depicted in the user's handwriting, as unstructured characters in the IVP, and as pretty-printed output after rendering.*

Participants were also able to correct recognition results and render the equation in the final form. However, correcting recognition presented some difficulties. First, the use of a separate input validation panel to correct character recognition followed by a rendering step to present structural analysis caused some confusion. This confusion resulted from a lack of understanding of the steps involved in recognition. Once character recognition was corrected, participants did not perceive the second recognition step, structural analysis, and did not understand why errors would occur once character recognition was completed. We are in the process of refining our recognition process to combine character recognition and structural analysis into a single recognition step. At a more detailed interface level, the use of multiple representations of the same equation – in ink, as recognized characters in the input validation panel, and as a mathematical expression after rendering (see Figure 6) – created a situation where users were unsure where to go to correct results. We saw several instances of users trying to correct the rendered expression rather than returning to their ink, or trying to scribble out and redraw in the input validation panel. Clearly, the separation of input, character correction, and rendering into three separate linear steps was not as intuitive as having a single, authoritative representation of the equation would have been.

The second area of difficulty in correcting recognition results involved transparency of errors. Often, particularly when structure was misrecognized, it was difficult for participants to perceive exactly why this was the case. This perception, however, is important, as it speeds the correction process. Participants would frequently make extensive corrections to an equation when only one erase and redraw might have corrected the results. This challenge with pen recognition systems is well known (see, e.g. [BLRZ02]), but how best to reveal the internal recognition processes is still an area for further research.

While extraneous ink, input validation, and the transparency of errors were all difficulties observed in our system, the system did allow all participants to successfully complete entry, recognition, and interaction for almost all of their equations during a thinkaloud style evaluation. The exceptions – idiosyncracies in character recognition, diffi-

culty with spacing, and a software bug – are being addressed through on-going research in recognition and error presentation and software bug fixes.

5.2. Mathematical Interactions

Mathematical interactions were a significant strength of our system. Context menus were a rapid, convenient way to make a large number of mathematical operations available to participants. Once the equation was recognized correctly, participants had few problems carrying out complex expansions, simplifications, plots, and matrix manipulations. One participant, an occasional user of a CAS, indicated that one of the strengths of this approach was the benefit of "not having to look up in Maple or Mathematica manuals how to do these" operations (P2).

Simlar to the difficulties with multiple representations observed during equation entry, there were occasional problems with multiple representations during interaction. For example, when users were asked to change their expressions, e.g. to change an exponent "4" to an exponent "42", they would frequently try to draw the "2" on the rendered expression rather than on their ink expression. We continue to work toward one authoritative representation of the equation which allows both input validation and mathematical interaction.

Unlike other pen-math systems, MathBrush allows users to interact with subexpressions, to transform these subexpressions, and to substitute the transformed subexpression back into the original equation. Participant P2 also noted the benefits of this, as working with subexpressions is "hard in Computer Algebra Systems" (P2). While the subexpression manipulation was more fluid than is typical in a CAS, which requires visual indexing of terms, it was still somewhat awkward for users. The use of a separate editing panel seemed unnecessarily tedious for users, though it did allow users to validate their subexpressions. Since this feature was positively received, we continue to explore designs that allow for both ease of selection and ease of subsequent manipulations.

Three participants commented favourably on our plot functionality. The one aspect of plots that did not seem intuitive was adjusting limits on the plot. Since editing a plot interactively with ink is not a commonly available feature, it was not immediately obvious to participants that such operations were possible.

6. Discussion

In this section, we will examine overall issues associated with the design of a pen-math system. We first focus on users adapting to the system to improve recognition. Next we discuss the user community for a pen-math system.

6.1. Users Adapting Their Behaviour

One interesting aspect of use of our system observed during our study was how readily users modified their writing to adjust to the tolerances of recognition. These adaptations included slowing down their timing, taking additional care with symbol spacing, and altering the way that they drew certain characters. With only 20 minutes of independent use, we noted a significant change in the way subjects drew equations.

The implications of adaptation are not immediately obvious. For example, while one participant noted that the "need to draw neatly" was something that slowed down use of the system (P3), particularly for input, any computer system to interpret math notation will have overhead associated with its use. While careful engineering and advances in hardware and software may speed this process, pen and paper are still both readily available and low in overhead. The question of whether the additional overhead associated with drawing neatly to enable recognition transforms MathBrush from acceptable to unacceptable overhead remains open, but this seems unlikely.

The strength associated with a CAS is that the required time to expand and simplify a high degree polynomial, for example:

$$(x^2+yz^3)^{42}-(x^3-y^2z^6)^{13} \tag{1}$$

is identical to the time taken to expand and simplify a smaller, more manageable polynomial, such as:

$$(x^2+yz^3)^2-x^2 \tag{2}$$

While the second polynomial is easily expandable by hand, the first polynomial is not in any reasonable time. The overhead of writing neatly and correcting recognition errors is clearly worth the time savings associated with expanding the polynomial using a CAS.

Pen-math systems occupy a space somewhere between pen and paper and CAS on the spectrum of ease of use versus computational power. Paper is a fluid, portable, convenient medium for writing information. Paper, however, provides no computational support for performing operations beyond those an individual can reasonably perform. CAS, on the other hand, have extensive computational support for complex mathematical operations, yet these systems require users to transform two dimensional math expressions into one dimensional strings prior to manipulation and to master a set of commands and/or menus to access the necessary operations.

A still open research question is where on this spectrum pen-math systems should be positioned. Should a pen-math system be an enhanced calculator that supports limited mathematical operations [XTh, vDO], or should it support the complex operations contained in a CAS? To answer this question, we revisit the issue of who the typical user of a pen-math system is.

6.2. User Community

In our system, we began our work by considering a pen-math system as an interface to a CAS. This resulted in the use of context menus to invoke a large set of mathematical operations, and the requirement that we limit the mathematical knowledge in MathBrush's front end.

The incorporation of a full-featured CAS into MathBrush is based on research in the use of computers in mathematics education. The premise of commercial systems such as Microsoft Math and MathJournal seems to be that there exists a difference between the math done by a CAS and the math required for high school or early university instruction. However, when Leinbach et al. studied use of CAS in high school mathematics classrooms, they noted that math instruction required access to many of the advanced CAS features, including expansion, factoring, substitution, subexpression manipulation, derivatives, etc. [LPE02]. Manipulation for comparison, conjecture, and the exploration of alternatives have also been pointed to as a benefit of CAS [LPE02], as the CAS assumes some of the mechanical processes involved in manipulating the same equation in multiple ways.

As noted earlier, the drawback of the use of CAS in classroom instruction is the need for extensive technical assistance to support students as they master use of menus, the location of commands, and the translation of syntax to provide input and interact with CAS [PHG04, Rut02]. In our study, participants familiar with CAS noted the use of context sensitive menus (which present operations based on an analysis of the identity of the equation being manipulated) as a benefit of our system. Users of MathBrush could examine the context menu to locate manipulations suitable for a given equation type. As well, the system preserves the two-dimensional representation while still allowing access to many of the necessary features of a CAS.

7. Conclusion

This paper describes the MathBrush system. We present background from education pedagogy that motivates our design decisions, and results from a thinkaloud study that validates our design and suggests areas for improvement. Together, these results provide guidance for the design of future pen-math systems.

Acknowledgements

This research was supported by Microsoft Canada, the Natural Science and Engineering Research Council of Canada (NSERC), and the Ontario Ministry of Research and Innovation.

References

[AK93] APTE A., KIMURA T.: A comparison study of the pen and the mouse in editing graphic diagrams. In *Proceedings of the IEEE Symposium on Visual Languages* (1993), pp. 352–357.

[Art02] ARTIGUE M.: Learning mathematics in a cas environment. *International Journal of Computers for Mathematical Learning 7* (2002), 245–274.

[BLRZ02] BLOSTEIN D., LANK D., ROSE A., ZANIBBI R.: User interfaces for on-line diagram recognition. In *Graphics Recognition: Algorithms and Applications, LNCS 2390* (2002), Blostein D., Kwon Y., (Eds.), Springer, pp. 92 – 103.

[CIPe01] CARLISLE D., ION P., POPPELIER N., (EDITORS) R. M.: Mathematical markup language (mathml) version 2.0. W3C Recommendation, http://www.w3.org/TR/2001/REC-MATHML2-20010221,.

[CY99] CHAN K., YEUNG D.: Recognizing on-line handwritten alphanumeric characters through flexible structural matching. *Pattern Recognition 32* (1999), 1099–1114.

[Lew82] LEWIS C.: *Using the thinking-aloud method in cognitive interface design*. Tech. rep., IBM T.J. Watson Research Center, Yorktown Heights, NY., 1982.

[LPE02] LEINBACH C., POUNTNEY D., ETCHELLS T.: Appropriate use of a cas in the teaching and learning of mathematics. *International Journal of Mathematical Education in Science and Technology 33*, 1 (2002), 1–14.

[LZ04] LAVIOLA J., ZELEZNIK R.: Mathpad2: a system for the creation and exploration of mathematical sketches. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004) 23*, 3 (2004), 432–440.

[Nie00] NIELSEN J.: Why you only need to test with 5 users. In *Jakob Nielsen's Alertbox* (2000). http://www.useit.com/alertbox/20000319.html.

[NL93] NIELSEN J., LANDAUER T. K.: A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems* (1993), pp. 206–213.

[PHG04] PIERCE R., HERBERT S., GIRI J.: Cas: Student engagement requires unambiguous examples. *Mathematical Education for the Third Millenium* (2004), 462–469.

[PS01] PIERCE R., STACEY K.: Observations on students' responses to learning in a cas environment. *Mathematical Education Research Journal 3*, 1 (2001), 28–46.

[Rut02] RUTHVEN K.: Instrumenting mathematical activity. *International Journal of Computers for Mathematical Learning 7* (2002), 275–291.

[SFUK03] SUZUKI M., FUKUDA F. T. R., UCHIDA S., KANAHORI T.: Infty- an integrated ocr system for mathematical documents. In *ACM Symposium on Document Engineering* (2003), pp. 95–104.

[Smi99] SMITHIES S.: *Freehand Formula Entry System*. Master's thesis, University of Otago, Dunedin, New Zealand, 1999.

[Tay95] TAYLOR M.: Calculators and computer algebra systems. *Mathematical Gazette 79*, 4 (March 1995), 6.

[vDO] VAN DAM A., OTHERS: Mathreco: Interpreting and correcting handwritten mathematics. http://graphics.cs.brown.edu/research/pcc/mathreco.html.

[XTh] XTHINK: Mathjournal. http://www.xthink.com/MathJournal.html.