Towards a Universal Interface for Real-Time Mathematical Communication

Marco Pollanen[1], Jeff Hooper[2], Bruce Cater[3], and Sohee Kang[4]

[1] Trent University, Canada
marcopollanen@trentu.ca,
http://euclid.trentu.ca/math/marco/
[2] Acadia University, Canada
jeff.hooper@acadiau.ca,
http://math.acadiau.ca/hooper/
[3] Trent University, Canada
bcater@trentu.ca,
http://www.trentu.ca/economics/staff_cater.php
[4] University of Toronto Scarborough, Canada
soheekang@utsc.utoronto.ca,
http://www.utsc.utoronto.ca/cms/sohee-kang

Abstract. Rapid growth in mobile computing has given rise to a dramatic evolution in communication tools that have the potential to transform the educational experience. Because of the complexities of mathematical communication, however, progress towards the realization of that potential has been particularly slow in the mathematical sciences. We are addressing this by developing iCE (interface for Collaborative Equations), a multimodal mathematical communication environment for post-secondary mathematics and statistics courses. Based on our experiences with this, we discuss the user requirements and difficulties in building a universal interface – one that allows users, ranging from mathematical novices to experts, to intuitively communicate mathematics in real-time using a broad range of computing devices.

Keywords: Mathematical Communication, Mathematical Collaboration, Mathematical User Interfaces, Formula Input

1 Introduction

Mathematical reasoning can rarely sustain itself in the mind alone. Just as Archimedes drew "geometrical figures in the ashes" [13], it is inconceivable that, in a traditional university classroom, a mathematical conversation could be sustained without the use of at least a chalkboard.

With the revolution in communication technologies, however, the face of post-secondary education is changing. Flipped classrooms, Massive Open Online Courses (MOOCs), hybrid and fully online courses use the Internet to provide students with access to learning materials from anywhere at any time.

Access to those learning materials is, of course, only one facet of the learning experience. While great libraries have existed for thousands of years, classroom education has flourished because of the importance of student-to-teacher, teacher-to-student and student-to-student communication. For example, outside-the-classroom student-teacher contact (e.g., office hours, e-mail) correlates positively with key educational indicators such as academic performance, student retention, and student satisfaction [9].

To remain relevant in the distance education market, it is therefore important for universities to be more than mere content providers – they must embrace the communication revolution to find ways to engage geographically-dispersed students, with greater immediacy and increased efficiency on large MOOC-like scales. There is evidence that higher education has fallen short in terms of the use of mobile devices for out-of-classroom contact. A recent study [4] suggests that 83.8% of students would use more app-based mobile communication if offered by instructors, and 81.1% wished that their professors offered more virtual communication options.

Anecdotal evidence suggests that virtual communication in mathematics lags significantly behind that in other disciplines. One reason is likely the lack of availability of user-friendly math-enabled communication tools. That is to say, while real-time communication tools such as instant messaging, chat-rooms and twitter are increasingly used by instructors to enhance the student experience, those tools are text-based, so their usefulness in mathematics is severely limited. Yet, the potential benefits of making mathematics communication more readily available are clear, for it has been demonstrated that online office hours can be very effective and efficient in mathematics courses, resulting in high levels of attendance and student engagement [5].

In this paper, we will discuss the user-interface challenges in creating a real-time communication environment for mathematics – one that is universal in that it makes casual communication, i.e. without the user needing to pre-install software or learn the interface in advance, between experts and novices alike as easy and intuitive as possible, regardless of whether they are using a more standard computing platform (Windows/Mac/Linux) or a mobile device such as a tablet or smartphone.

2 Mathematical Input

To achieve the goal of having mathematical formulae be composed and distributed in real-time, there are several issues that must be overcome when designing the user-interface and software.

First, there is the symbol problem. Mathematics makes use of far more symbols than can be conveniently laid out on a keyboard. Expansion of what a keyboard can do through the use of key combinations using the “alt”, “ctrl” or “command” keys is difficult to achieve in a platform- or device-independent way (e.g., these key-sequences are not available on mobile platforms). Even where this expansion can occur, its use is difficult for a novice to learn.

Second, there is the layout problem. Many mathematical expressions are written in a 2-dimensional way, with symbols appearing above and below other symbols, and where not all symbols are of the same size. The relative positioning of the symbols is often important: altering those positions can change their meaning in a critical way.

Third, real-time input of mathematics also presents a problem with sequencing. The order in which mathematical symbols are transferred to paper or screen often does not reflect the left-to-right order of the final expression.

Fourth, while typos in real-time text-based communications are mostly harmless (the reader can typically figure out the intended meaning from the surrounding text) mathematics is a very precise language in which typos lead to errors in mathematical expressions or to confusion among users.

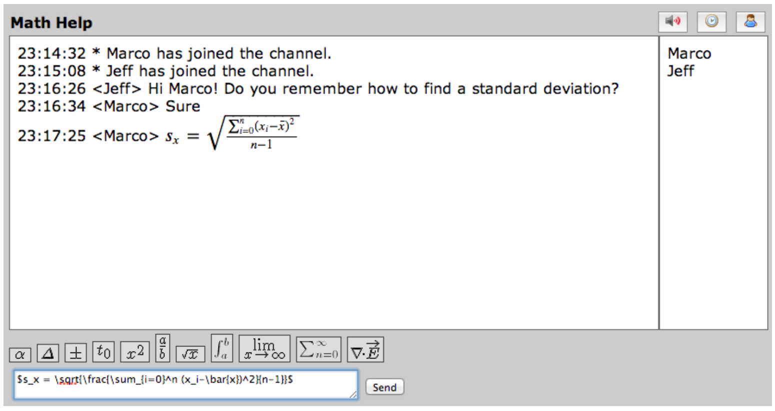
3 Choice of Interface / Platform

Recent years have seen the fragmentation of platform monoculture. There is, for example, a trend in many workplaces away from the support of a corporate standard towards encouraging employees to 'bring your own device.' A similar trend can be seen in post-secondary education, where many universities that once provided a standard notebook computer to every student, now facilitated by the advent of campus-wide wifi, expect students to bring their own.

A decade ago, it was a safe bet that every student had access to a desktop/notebook running Internet Explorer. Today, the landscape is more varied, with devices, such as smartphones, tablets (some with a stylus; some without) and a resurgence of Macbooks – a wide assortment of available devices that have vastly different user interfaces, screen sizes and computational power. With the introduction of Chrome, Firefox and mobile browsers, the market share for Internet Explorer has decreased significantly.

For a mathematical communication interface that is meant for novice users – for example, students wishing to only casually drop into an online office hour – the requirement of special software or hardware, or the need to download and install a specialized interface, may represent a barrier. A browser-based interface is, therefore, a natural choice, provided it allows for intuitive behaviour on multiple browsers. While there has been a proliferation of computing platforms, many mainstream applications, such as Google Docs and Office 365, are already browser-based and the web browser is quickly becoming a standard interface for full-featured applications, especially collaborative ones. Rich browser-based applications used to rely on Java or Flash, but these languages are no longer well-supported and are not viable options.

Xpress [15] is a prototype for a new equation editor model which allows users to build expressions by placing drag-and-drop symbols anywhere in their expression they wish without structural constraints. It was developed as a browser-based application around a Scalable Vector Graphics (SVG) document supported by Asynchronous Javascript and XML (AJAX). In developing the communication interface iCE, Mathematical Markup Language (MathML) was considered



```
23:14:32 * Marco has joined the channel.
23:15:08 * Jeff has joined the channel.
23:16:26 <Jeff> Hi Marco! Do you remember how to find a standard deviation?
23:16:34 <Marco> Sure
```

23:17:25 <Marco> $s_x = \sqrt{\frac{\sum_{i=0}^n (x_i - \bar{x})^2}{n-1}}$

Marco
Jeff

```
$s_x = \sqrt{\frac{\sum_{i=0}^n (x_i-\bar{x})^2}{n-1}}$
```

Fig. 1. MathIM

as it is intended to be the standard for mathematics on the Internet. However, at the time of planning for iCE it was not fully supported [1], so iCE has been developed around a front-end of Javascript/SVG. Mathematical communication involves more than the input of formulae, and SVG makes implementation of diagramming tools, scaling, and cut-and-paste easy and provides a default document format.

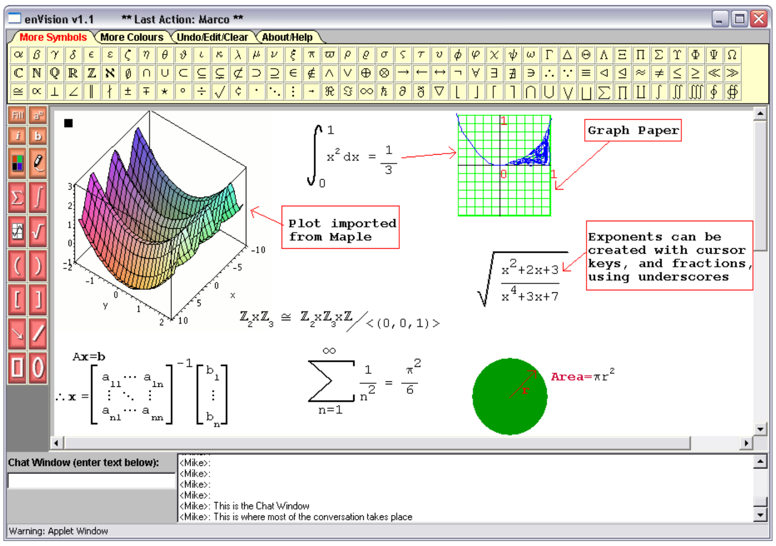
4 Existing Math Communication Tools

To date, few communication tools have been developed specifically for mathematics communication. Two freely available examples that we have experience with are *MathIM* and *enVision*:

MathIM: MATHIM is a typical Javascript/HTML-based scrolling chat with the added ability to insert TEX images by including TEX code between `$...$` delimiters. There are a small number of template buttons that insert some TEX-code into the chat input line to assist with input. (See Figure 1).

Our experience is that TEX input is very difficult for even advanced mathematics undergraduates and is rarely used by students. Thus students using MathIM tend to write their expressions descriptively using only text, which can lead to many ambiguities. The software also only allows one line at a time, with no ability to edit previous expressions. The ability to interact with and edit expressions would speed up communication (e.g., a professor at a chalkboard erasing symbols that are cancelled in an expression without rewriting it).

enVision: ENVISION is a typical Java-based shared whiteboard application that includes a chat window for textual conversation. Its whiteboard is modelled after a raster-based paint program which would allow users to either draw with a pen or drag-and-drop or insert resizable mathematics symbols anywhere on the



<Mike>:
<Mike>:
<Mike>:
<Mike>:
<Mike>: This is the Chat Window
<Mike>: This is where most of the conversation takes place

Fig. 2. enVision

shared canvas. An audio communication mode has been produced as a separate add-on. (See Figure 2).

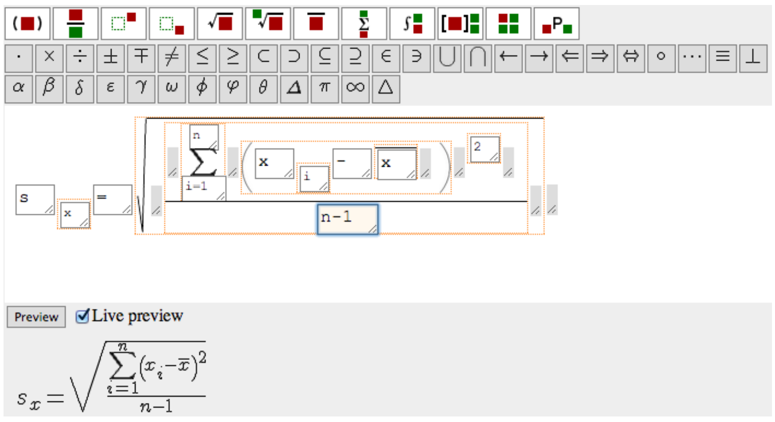
Our experience with this application is that students are able to communicate relatively easily with this type of interface. However, since it is pixel-based, deleting is accomplished by making use of an eraser brush, and symbols on the canvas cannot be easily selected and moved like in a vector-based diagram editor. More importantly, Java is not supported on newer devices, such as tablets, and hence enVision is not a universal option.

Other technologies, including Skype and screen sharing could be used for mathematical communication, but these approaches do not directly solve the problem of joint work on shared equations and do require the pre-installation of software which might represent a barrier for a student who has a simple immediate question.

5 Models for Equation Editing

There are four distinct User Interface (UI) models for the creation of equations on a computer or device:

Structure-based: Graphical structure-based editors are the most common form of mathematical input UI. These editors typically separate mathematical structures, such as fractions, from the individual symbol elements. To enter a complicated expression requires the user to first set up an appropriate structure of nested boxes, which may then be populated with numbers or symbols (see Figure 3). These structures can be difficult to navigate and manipulate, and there are



$$s_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Fig. 3. Structure-Based Editor: BrEdiMa [10]

inconsistencies between editors [12]. One study [3] found that these editors are difficult for students to use – there may be a steep learning curve in that they force students to mentally parse a full expression first and in many cases enter the expression in an unnatural order. This extra cognitive load can lead to dual-task destructive interference with the main task of communicating mathematics [11].

Pen-based: A robust pen-based mathematical expression input system that allows one to input expressions as easily as one can on paper could be described as the "holy grail" of mathematical equation input research. Unfortunately, the challenge of recognizing handwritten mathematical expressions is much more complex than that of recognizing handwritten text. While characters in handwritten words have a one-dimensional layout (i.e., are arranged linearly in order), mathematical expressions often have a two-dimensional layout (e.g., fractions, matrices). Mathematical symbols also tend to be very similar, so determining the differences between 1.2 and $1 \cdot 2$, between $<$ and $\langle$, or distinguishing among the symbols $\cdot, \bullet, O, o, 0, \ ^\circ, \circ, \odot, \oplus, \otimes, \emptyset, \oslash, \phi, \ominus, \theta, \Theta, \ldots,$ is a very difficult task – one of many UI challenges remaining with mathematical pen input.

For the purposes of creating a universal user interface model, pen-input alone is not a sufficient approach. Although pen-based devices have been available since the 1980s [17], they are still not commonly used. While touch interfaces, such as the iPad, would allow a user to draw an equation with a finger, the low touch precision of these interfaces means that the size of the equation must be large in proportion to the screen area.

Text-based: Mathematical content can certainly be communicated using text only, but plain text communication is a poor option for mathematics beyond the simplest expressions. An option that is sometimes used to address this is TEX, the technical writing language created by Donald Knuth in 1978. TEX was designed for "the creation of beautiful books" [6], and its relatively steep learning curve limits its use to experts, rather than to casual users. For example,

```
s_x = \sqrt{\frac{\sum_{i=0}^n (x_i-\bar{x})^2}{n-1}}
```

is a LaTeX representation for the elementary formula

$$s_x = \sqrt{\frac{\sum_{i=0}^n (x_i-\bar{x})^2}{n-1}},$$

which defines the sample standard deviation studied in high school and university statistics. The LaTeX string is not intuitive enough for novices.

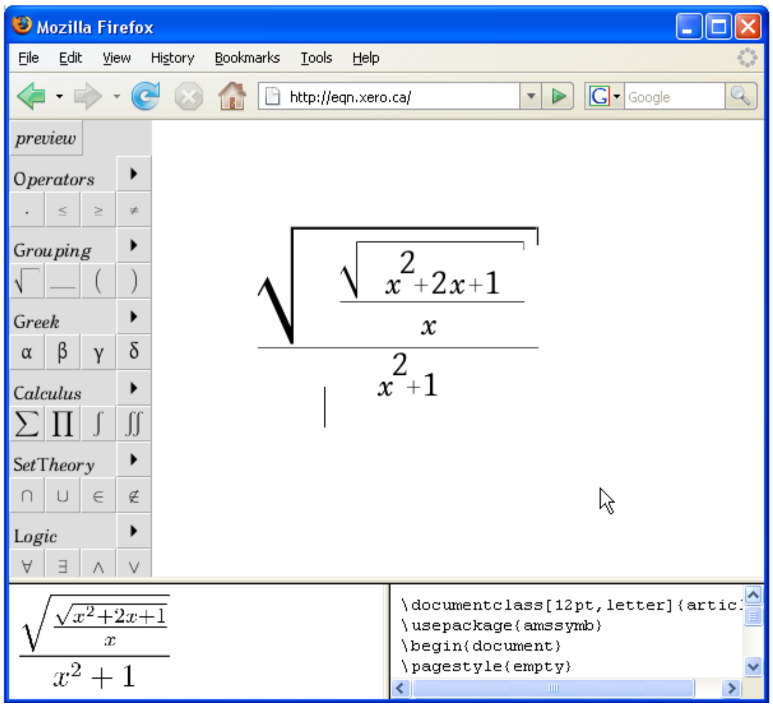
Unconstrained Editor: Xpress [15] is an unconstrained editor where users are free to create an expression any way they wish by essentially "drawing" it on a canvas (see Figure 4). However, it is different from enVision in that the UI model is that of a diagram editor, so the mathematical symbols are resizable/draggable elements. Once an expression is drawn, Xpress applies a spatial recognition algorithm to recognize it. See: `http://www.xero.ca/xpress.swf` for a Flash video demo of Xpress to better understand this approach. In the main pane, a user draws a diagrammatic representation of the expression and it is recognized and converted to LaTeX code (lower right in the video). The LaTeX code is compiled and the result is previewed (lower left in the video) to demonstrate correct recognition.

It has been shown that with this unconstrained approach, students tend to write expressions in the same order that they would on a piece of paper and with greater speed than with a structure-based editor [3,16].

6 Specific Universal Interface Model

The goal of iCE is to develop a multi-user communication interface that will run on a wide range of platforms and that will allow users with different degrees of mathematical experience to communicate with each other in a casual (but accurate) manner. We will consider a mathematical UI model based on an unconstrained approach, such as with enVision/Xpress, but it will include as many elements as possible from other approaches with which a user might be familiar, including pen, structural and TeX.

Until recently, the most common platform for accessing the Internet was a personal computer equipped with a keyboard and mouse (or equivalent pointing device). In recent years, with the advent of mobile computing, touch devices, such as smartphones and tablets, are becoming ubiquitous. These devices typically have small screen sizes and rely on virtual onscreen keyboards. Some smartphones and tablets have a built-in digital pen, but others, such as iOS devices, have no built-in pen. Even when a digital pen is available, these styli are blunt and do not provide a high degree of accuracy on a small screen. Thus, to be usable on all platforms, a universal interface must consider the UI models for keyboard, mouse, pen, and finger touches.



```
\documentclass[12pt,letter]{artic
\usepackage{amssymb}
\begin{document}
\pagestyle{empty}
```

Fig. 4. XPRESS

6.1 Keyboard Interaction

There are three basic interactions in the iCE model: a text mode, a symbol mode, and a TEX mode.

Text Mode: As iCE is designed as an extension of the diagram editor UI model, it includes a text mode which can be engaged from the mode selection panel. The text mode allows a text box to be placed anywhere on the screen with text that can assume various font attributes: style, size, italics, bold, colour, etc. It behaves similar to the text mode found for labelling diagrams in a typical diagram editor.

TEX Mode: TEX Mode is similar to Text Mode in that text is entered into a text box. However, in this case, once the text entry is complete, the TEX is compiled into SVG using MathJaX via a web-service call. At this time, the TEX input cannot be edited after it is compiled, but the resulting visual expression can be edited using the iCE drag-and-drop UI model.

Symbol Mode: This UI mode is unique to the iCE/XPRESS model. In the iCE model, when in almost any mode, there is a cursor on the screen. Whenever a user clicks or taps on a blank section of a canvas, the symbol mode cursor is placed at that point. Cursor keys also allow the cursor to be moved freely in any direction in fractional steps. Once a key is pressed on the keyboard, that key's corresponding symbol is placed in the cursor location. With repeated presses of the key, however, the placed symbol cycles through several logical alternatives. For example, "$<$" cycles through "$<$", "$\leq$", "$\ll$", "$\angle$", while pressing "A" cycles through "A," "α," "$\forall$," "$\wedge$," "$\aleph$," "$\angle$." If the symbol mapping is not intuitively obvious, a symbol could be inserted by clicking on a symbol from a palette.

In addition to these keyboard modes for the shared SVG document, there is also a chat pane. In our model, the shared SVG document is like a physical chalkboard where individuals are engaging, while the chat pane is for the bulk of the conversation. While this is primarily text based, it will support symbol input from palettes as well as TEX for expert users as MathIM does.

Text interactions work well on a desktop or laptop computer. However, touch devices typically have virtual keyboards. While a custom virtual keyboard could solve the symbol problem by offering alternate key selections that have math symbols, this can be problematic: Android allows the user to install custom keyboards, but iOS does not. However, custom Javascript keyboards can still be implemented in a web browser.

Virtual keyboards also have several drawbacks. First, they can occupy about half of the screen when engaged. This is further compounded by the fact that browser applications are embedded in browsers, which typically have their own UI components that can occupy their own screen space. Second, because precision in determining touch location is low, it is easy to make errors on touch keyboards. Text methods for input on touch devices often include correction – for example, dictionary matching – to increase accuracy. For mathematical expression input, the very broad range of correct mathematical expressions makes robust corrective algorithms extremely difficult to implement. Third, the keyboards can be very distracting as they repeatedly appear/disappear as the user switches between text input and a touch interaction, such as a click or drag.

For touch devices, the iCE model currently uses a combination of OS virtual keyboards (text mode or chat window) and a basic custom javascript virtual keyboard (symbol mode). More research needs to be conducted concerning the best way to handle text input of mathematics on a virtual keyboard. For example, the iOS virtual keyboard is not well suited for inputting TEX, as symbols, such as `{`, `}`, `$`, `\`, `_`, are not grouped together or found on the same screen.

6.2 Mouse/Pointer Interaction

The mouse/pointer interaction in iCE is modelled after that of a standard diagram editor, where objects may be placed anywhere on a canvas, moved and resized. In the case of iCE, instead of objects being restricted to diagram elements, such as boxes and arrows, mathematical symbols may be placed anywhere on the canvas. These symbols may be manipulated as follows:

Moving: Any individual symbol may be moved by just dragging it.

Selection: Any group of symbols may be selected by dragging out a selection box starting from any blank spot on the canvas (see Figure 5). When completed,

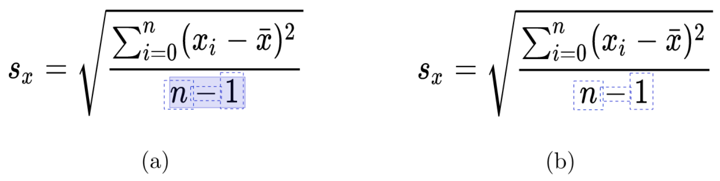


Fig. 5. Selection in iCE: (a) selecting a subset of symbols; (b) a single group of the selected symbols are outlined.

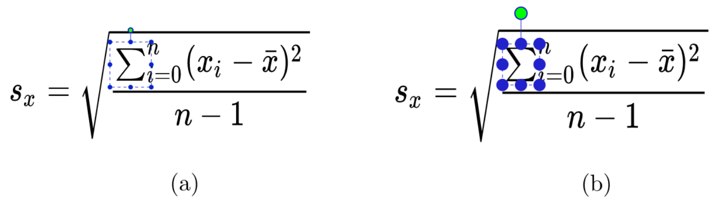


Fig. 6. Resizing in iCE: (a) a resize widget on a PC; (b) a larger resize widget on an iPad.

any objects falling inside the box will be selected. A selected group can be moved by dragging the group. By clicking/tapping on a blank selection of the canvas, the group is unselected. Currently, iCE implements rectangular selection for all devices, although our tests indicate that, particularly for touch devices, a lasso-style selection of elements, in which users circle the group of elements that they want to select with their finger, may also be an effective possibility.

Resizing: If an object is clicked on or tapped on, resizing widgets appear which, when dragged, resize the object. These resize widgets will automatically be larger on touch devices to account for the lower click/tap precision. (See Figure 6.)

6.3 Expression Recognition

A selected group of symbols may also be converted into TEX typeset output form for high quality visual display by having the user click on the "convert to TEX" button on the mode panel, using a similar approach to the expression recognition found in XPRESS.

iCE also has a free sketch mode, where a user may draw freely in a layer of digital ink. In the future, once a group of pen-drawn symbols is selected, the group might be passed to a handwriting recognition algorithm to have the entire expression to convert into TEX typeset output.

At the moment, expressions are only recognized for presentation structure. In the future, it might be possible to try to recognize mathematical expressions for the purpose of eliciting mathematical content structure. Once recognized, the extracted content could be passed to a Computer Algebra System to assist with intermediary steps in order to speed up communication.

Inline expression recognition could also be valuable for assisting users in forming expressions. In its current form, iCE has a "snap" feature, which, when dropping or resizing symbols, forces the symbols to stick to some invisible lines. As a possible extension, an inline expression recognition algorithm could, as an expression is being created, determine the most likely locations for additional symbols to be appended, and, by way of a magnetic attraction, provide the user with subtle hints for where the additional symbols should be placed. Analogous to the auto-complete feature now becoming ubiquitous in many text applications, these suggestions could then be used or ignored – the user would still be free to place the symbols anywhere on the canvas. This approach can also be thought of as a hybrid between an unconstrained editor like XPRESS and a more structure-based editor.

7 Conclusion/Discussion

In this paper, we have considered a model for real-time communication that combines UI elements from palette-based structural editors, keyboard-based TEX input, whiteboard UIs and text messaging panels. The goal is that each user would be able to use the model or models that best suit their experience and hardware limitations, making communication as easy as possible.

Our experience is that this model works very well on desktops/laptops. The drag-and-drop portion of the model also works well on tablets; virtual keyboard improvements would still need to be explored. Because of its limited screen size, the smartphone provides a challenge for inputting equations quickly and a robust zoom in/out feature will likely have to be developed. However, the high resolution of smartphone screens means this model allows the user to easily view the discussion and provide occasional input. It should also be noted that, if the trend towards increasing smartphone screen sizes continues, these concerns may be mitigated.

When developing a mathematical communication interface, careful consideration must also be given to the issue of conflict resolution. Is the canvas truly shared, as in a free-for-all, or should there be a method of managing which user has the canvas? How should the editing of other users' contributions be managed? What about sequencing – if, say, a user arrives late, can he/she play back a portion of the conversation? These questions, while important, are beyond the scope of this paper.

There is undoubtedly a need for the development of communication environments designed specifically for mathematics – a task made more difficult by the growing range of user devices. We hope that, by exploring this issue, we are taking the first steps towards building a universal interface for the real-time communication of mathematics.

References

1. Davide Cervone, *MathJax: A Platform for Mathematics on the Web*, Notices of the AMS 59, No. 2, 2012.
2. Richard Fateman, *How can we speak math?* Unpublished manuscript (2013): `http://www.eecs.berkeley.edu/~fateman/papers/speakmath.pdf`.
3. Davood G. Gozli, Marco Pollanen, and Michael Reynolds, *The Characteristics of Writing Environments for Mathematics: Behavioral Consequences and Implications for Software Design and Usability*, Lecture Notes in Artificial Intelligence 5625, Proceedings of MKM 2009, Grand Bend, Ontario, Canada, pp. 310-324, 2009.
4. Lora Helvie-Mason, *Office Hours: 'There's an App for that?!' Student perceptions of faculty channels for out-of-class communication*, International Journal of Instructional Technology and Distance Learning, Vol. 9, 2012.
5. Jeff Hooper, Marco Pollanen, and Holger Teismann, *Effective Online Office Hours in the Mathematical Sciences*, Journal of Online Learning and Teaching, Vol. 2, No. 3, 2006.
6. Donald E. Knuth, The TEXbook. Addison-Wesley, Reading, 1984.
7. MathJax homepage: `http://www.mathjax.org`
8. Robert Miner, *The importance of MathML to mathematics communication*, Notices of the AMS 52, No. 5, 2005.
9. Marjorie K. Nadler and Lawrence B. Nadler, *Out-of-class communication between faculty and students: A faculty perspective*, Communication Studies, Vol. 51, No. 2 (Summer), pp. 176–188, 2000.
10. Yasuhito Nakano and Hirokazu Murao, *BrEdiMa: Yet Another Web-browser Tool for Editing Mathematical Expressions*, in: Proceedings of MathUI 2006, 2006 (online).
11. Sharon L. Oviatt, Alexander M. Arthur, Yaro Brock, Julia Cohen, *Expressive pen-based interfaces for math education*, CSCL 2007: pp. 573–582, 2007.
12. Luca Padovani and Riccardo Solmi, *An Investigation on the Dynamics of Direct-Manipulation Editors for Mathematics*, Lecture Notes in Computer Science, pp. 302–316, 3119, 2004.
13. Plutarch, The Lives Of The Noble Grecians And Romans, Vol. 1, Dryden translation, Modern Library, New York, 2001.
14. Marco Pollanen, *Interactive Web-Based Mathematics Communication*, Journal of Online Mathematics and its Applications, Vol. 6 , No. 4, 2006.
15. Marco Pollanen, Thomas Wisniewski, and X. Yu, *X: A Novice Interface for the Real-Time Communication of Mathematical Expressions*, Proceedings of MathUI 2007, Linz, Austria, June 2007.
16. Marco Pollanen and Michael Reynolds, *A Model for Effective Real-Time Entry of Mathematical Expressions*, Research, Reflections and Innovations in Integrating ICT in Education, Formatex, 2009.
17. Ben Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 2nd Ed. Reading, MA: Addison-Wesley.
18. Ernesto Tapia and Raúl Rojas, *Recognition of On-Line Handwritten Mathematical Formulas in the E-Chalk System*, Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 980–984, 2003.