Networked Digital Whiteboard with Handwritten-Symbol Interpreter and Dynamic-Display-Object Creator

Atsuhide Kobashi
Henry M. Gunn High School
Palo Alto, California

Abstract

We present a unique Tcl/Tk-based whiteboard system that facilitates discussions among multiple participants located at remote sites as well as in classroom or conference-room setups. One of its uniquenesses is its capability to convert handwritten complex-structured math/scientific symbols to clean custom-font-based presentations. Another uniqueness is that it enables the user to easily create dynamic discussion tools incorporating various widgets instantly during a discussion, and send them to the whiteboard server for display and manipulation. The above features can be easily augmented to fit various domains of discussion by adding more Tcl scripts and/or C modules. All of these unique capabilities are made possible by Tcl's easy mergeability with C and its superb scripting characteristics.

1. Overview

We often felt a need to have a whiteboard-like presentation tool that facilitates easy two-way interactions on the projected screen among all the participants in a room. Conventional presentation tools are designed to only facilitate one-way information flow from the presenter to the audience. We created our new whiteboard to break this limitation and to enable all the participants in a classroom or convention hall to print characters, draw figures, show images, and even display widgets to manipulate on the projected common display.

Although initially our goal was to create a two-way presentation tool for classroom and conference setups, we further enhanced our system to a full-fledged networked whiteboard so that it enables any remote networked party to view the whole whiteboard on their own display and participate in the two-way whiteboard-based discussion.

The system is built upon Tcl's well-designed and easy-to-use network tools. As delineated in Figure 1, the presenter and the participants are connected by a network.

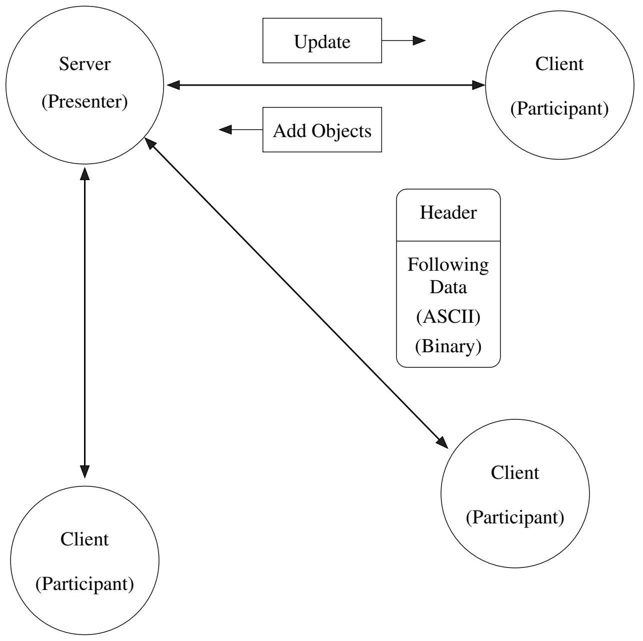


Figure 1. Networked Presentation Scheme

The particpants can augment the presenter's screen with widgets, texts, or other materials to expand the discussion. Once a new addition has been made to the presenter's screen, the server will update each of the participants' screens to reflect the newest state of the discussion.

These interactions between the participants and the presenter require multiple types of data to be sent over the network, including ASCII and binary data. In order to efficiently transfer such information, we are attaching headers as shown in Figure 1, to indicate the types, lengths, and handling methods for the data, just like many network transaction schemes such as TCP/IP [4], although our header's descriptions are of a higher level than those underlying network schemes.

The whiteboard also takes advantage of Tcl's unique and powerful multiple interpreter capability [5] to safely allow multiple participants to display characters, images, and widgets on the common whiteboard without interfering with the other participants actions.

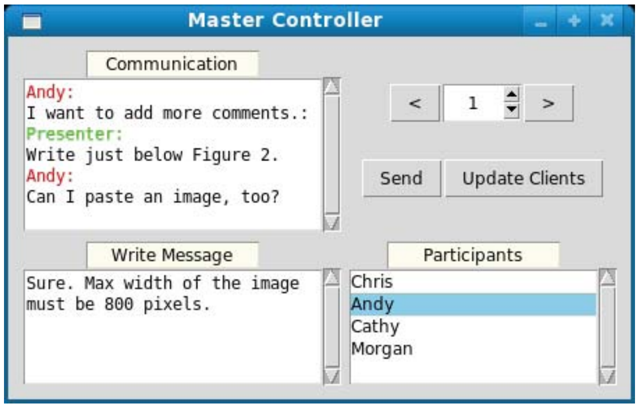
The objects transmitted through the network are Tcl scripts and their accompanying data which create displayed objects on the whiteboard being interpreted by each Tcl interpreter. The transmission of the dynamically created widgets by the Widget Creator (See Figure 6) is achieved, as explained below, by sending only the data given in the Widget Creator, without sending a Tcl script.

Our new whiteboard has two unique capabilities that enhance the contents of the whiteboard beyond any conventional whiteboard's: Handwritten symbol interpretation/conversion and dynamic-display-object creation. One of the functionalities we felt necessary while conducting discussions on mathematics and scientific subjects with conventional presentation tools is the capability to create special complex symbol structures quickly and easily during a presentation. Examples of such complex 2-D layout symbol structures include sigma, integral of calculus, limit, etc. which have many parts laid out in a 2-D area. Their structures can be more complex than what the usual word processors' templates can handle.

To solve this problem, we developed a module that is capable of converting handwritten (by mouse or pen tablet) special symbols to custom-font-based clean printings, as handwriting is the quickest and easiest way of specifying a symbol structure. Moreover, every part of the created clean printings is easily relocatable to fit any special layout need.

Another feature that we developed and is currently nonexistent in any other presentation tool we know of, is the ability to quickly and easily add dynamically manipulable objects to the whiteboard during a discussion. Our new whiteboard enables the participants to send a set of widgets that other users can operate on the whiteboard, to generate dynamic results. They can create these widgets via a Tk-based GUI. For example, the user can send a graph linked to a formula whose variables can be adjusted with spinners. The generated dynamic results are displayed as the updated graph in the widget which the user added to the common presentation screen.

Our system is written mostly in Tcl and only those modules which need high speed processing (e.g. handwritten symbol interpretation) are written in C and embedded [3] in the system by taking advantage of the Tcl's easy mergeability with C modules. Unlike monolithic systems such as Word and OpenOffice, our system can easily be augmented to fit the users' various needs by taking advantage of Tcl's easy scripting and incremental augmentability. For example, the special mathematical/scientific symbol interpreter can be enhanced by adding more C modules to add more interpretable symbols. The dynamic-display-object creator can also be easily augmented by adding more Tcl script templates.



Andy:
I want to add more comments.:
Presenter:
Write just below Figure 2.
Andy:
Can I paste an image, too?

Sure. Max width of the image must be 800 pixels.

Chris
Andy
Cathy
Morgan

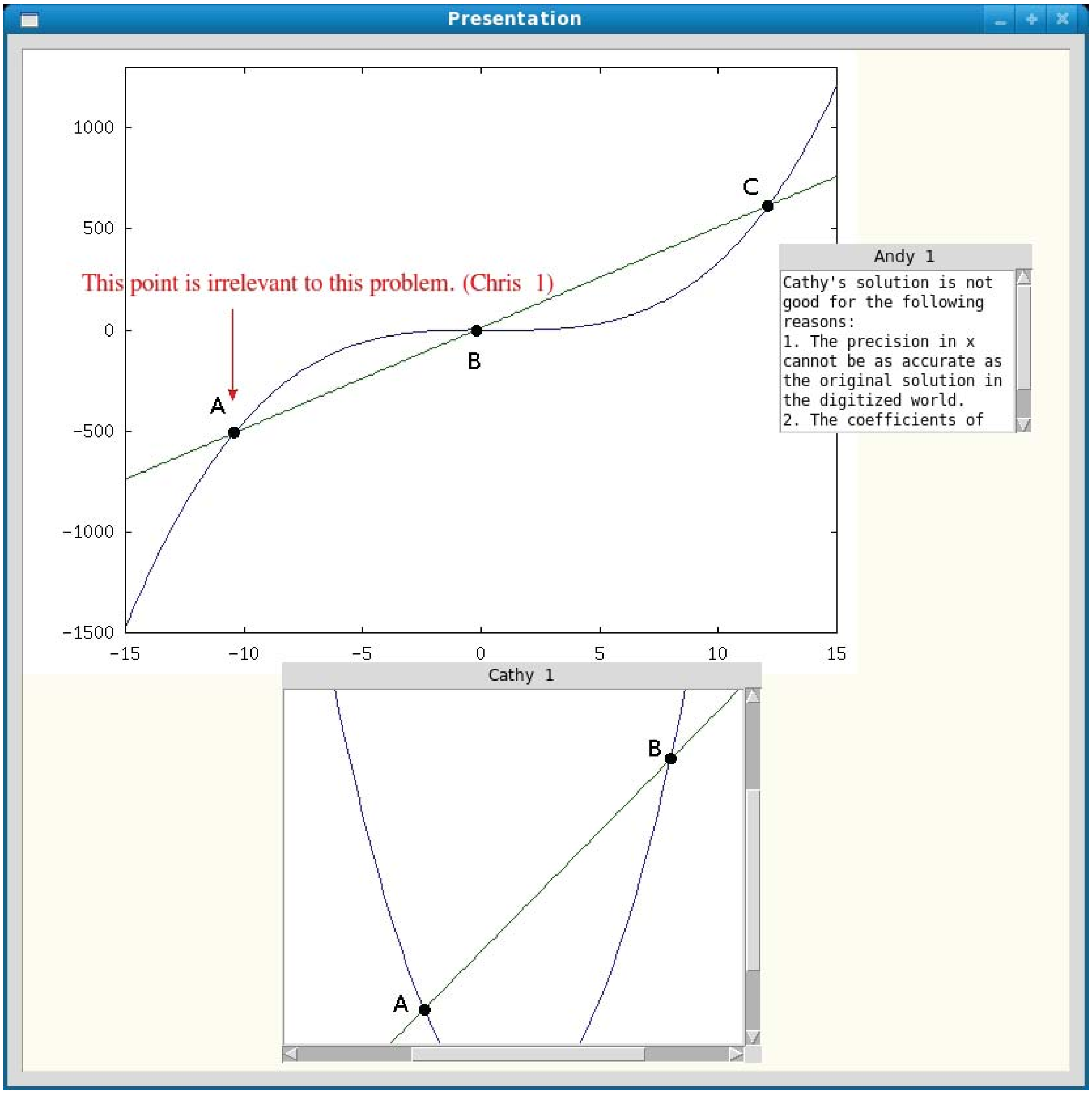
Figure 3. Master Controller

2. Operation

In this section, we present the functionality and usage of each of the system's accompanying tools.

Figure 2 displays the presenter's (server) whiteboard augmented with widgets and texts which convey the ideas of both the presenter and several other participants. The background with the large graph is the original form of the screen presented by the presenter, to which the participants are responding by adding objects which they created during the discussion. All participants can transmit their ideas via direct embedding of text, as displayed by the red comments around the graph in this example. Our system also provides participants with the capability to send and insert widgets onto the presenter's whiteboard, as exemplified by the embedded windows titled with the participants' names and target IDs. (The target IDs are used when spinboxes are linked to the widgets to identify the connected widgets.)

The fundamental tool for the presenter is the the Master Controller shown in Figure 3. This tool provides the presenter with the ability to communicate with the participants through the communication windows, and control the discussion by changing the displayed page on the screen. The presenter can advance/backup the page one by one at each press of the big arrows, or jump to a remote page using the spinner arrows in the spinbox. The communication windows allow the presenter to quickly converse with select participants in a troubleshooting or moderating role. While this may be unnecessary in small conference or classroom setups, it is particularly convenient when holding discussions with remote participants. The presenter's messages can be addressed to a select participant, but the conversation can be viewed by all parties. Each participant also has a similar tool to this Master Controller, albeit without several capabilities including updating and selecting pages.

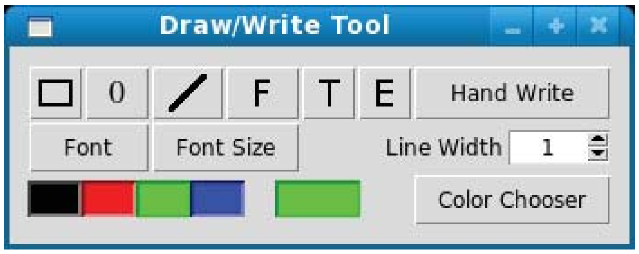


Cathy's solution is not good for the following reasons:
1. The precision in x cannot be as accurate as the original solution in the digitized world.
2. The coefficients of

Figure 2. Presentation Screen

Figure 4 displays the Draw/Write Tool available to all participants as well as the presenter. It provides the functionality for creating both graphic and text objects, including the capability to create complex symbols from handwriting. From left to right, the top row of buttons provides the standard capabilities for drawing rectangles, ovals, and lines, free-hand drawing, inserting texts, and erasing. The "Hand Write" button opens a new "Write" window (Figure 5) in which the user can draw complex symbol structures to be converted into a clean font. Once the handwriting is completed, the user can press the Convert button to print a clean form of the symbol at a position designated by a previous mouse click.

The Widget Creator tool shown in Figure 6, enables the user to create three types of widgets: texts, canvases, and spinboxes. Dimensions can be specificed for text and can-



Line Width 1

Figure 4. Draw/Write Tool

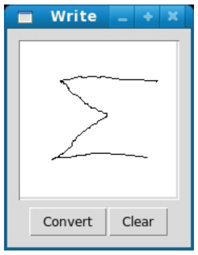
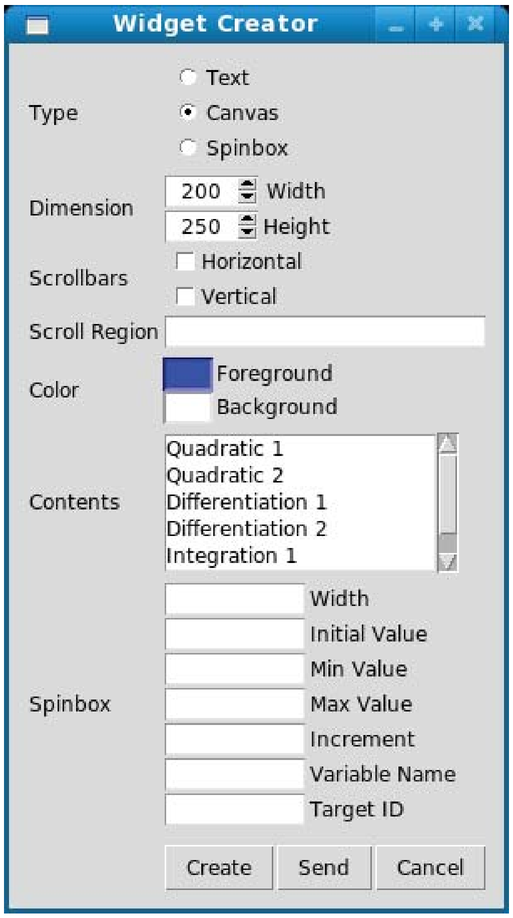


Figure 5. Handwriting Entry Window



Type: Text [✓ Canvas] Spinbox

Dimension: 200 Width, 250 Height

Contents:
Quadratic 1
Quadratic 2
Differentiation 1
Differentiation 2
Integration 1

Figure 6. Widget Creator

vas widgets. The canvas type has the option of a Scroll Region which the user can set by inputting four coordinates: Xmin, Ymin, Xmax, Ymax. The user can also determine the contents of the text or canvas widget by selecting a pre-made content they prepared beforehand, such as graphs, charts, images, text or a combination of them. The user can also create the contents on the spot, using the Draw/Write Tool. Participants can attach spinboxes to accompany these charts/graphs, such that the values of certain variables can be dynamically altered and the results of the updated values are reflected on the graphs/charts displayed in the widgets.

3. Underlying Mechanism

For the network communication we took advantage of the Tcl's well designed networking tools for a quick and easy development. They are far easier to handle than their C-based counterparts. For this purpose we used the Tcl commands such as socket, fconfigure, fileevent, and read. These commands aided in the smooth setup of a server-client configuration on which our system is build upon.

Although the forms of the data that is sent over our network are of a higher level than those of the underlying networking schemes, we need to handle multiple types of data with different objectives. The data types constitute of Tcl scripts, other ASCII data (dynamically created widgets' specification transmitted from the Widget Creator, contents of text widgets) and binary data (images). For efficient handling of these multiple types of data, we used a header that contains detailed information on the data following the header. The header provide the information on the type, length, and handling method for the data in connection to the related Tcl scripts. This strategy helps optimizes the efficiency of the data transmission through the network.

One of our unique capabilities of sending dynamically created widgets, is efficiently accomplished through the transmission of only the necessary data, instead of entire Tcl scripts for creating the widget. These data include specifications such as color, dimensions, and contents, which are inputted by the user through the Widget Creator tool. This scheme is made possible because the presenter's (server) side is designed with the capability of creating and embedding entirely new widgets with only these specifications. This strategy substantially reduces the network traffic as compared with sending Tcl scripts, and provides a smoother interaction between the participants. The same strategy is used when the clients' screens are updated by the presenter.

The algorithms utilized in our hand-written symbol and structure interpreter are highly complex and were derived from papers and other sources on computer vision. Through this development process, we learned that even at the forefront of computer vision research, there is no perfect interpreter for the types of complex symbols we require. Also, as a result of the high computation load involved in this interpretation process, we wrote the processing modules for this task in C. This modular system provided us with speed as well as easy augmentability, as discussed earlier. The true value of this interpreter surfaces in situations requiring special symbols unavailable in most fonts, or complex nesting which cannot be accomplished by most word processors. Currently, the fonts for these successfully converted symbols are generated by Tex.

As for the handwritten symbol interpreters, we found that there are plenty of algorithms available for our purposes [1, 2]. They are generally very sophisticated and require high computing power. However, we learned during the development of this system that there is no handwritten symbol interpreter as good or near a human's capability even at the forefront of the computer vision research. Therefore, we still claim the state of our system with respect to this capability as "still under development".

For the set of symbols that are selected for the current version of our system, the clean fonts to which the handwritten symbols are to be converted, are results created by Tex. In the future, once we try to handle symbols unavailable even from Tex, we will need to create fonts by ourselves.

All of our system's GUI as well as the dynamic widget creation functions provided to the user through the Widget Creator are implemented in Tk. We took advantage of Tk's easy programming characteristic and abundant features and functionalities, to develop the robust user interface for our system. Other advantages of Tk are that the code is light weight relative to the functions it can perform and the scripts are human readable which enables easy inspection by the developer. These characteristics make Tk a perfect choice for the transmission of graphic objects, especially dyamically created ones. Furthermore, Tcl provides the valuable functionality of multiple interpreter capability, to our endeavor, as each client's input onto the presenter's screen can be processed in independent slave interpreters, so as not to interfere with other parties' operations.

4. Problems and Future Plan

As mentioned earlier, our handwriting interpreter is still under development and its capabilities are currently limited to interpreting the mathematical symbols of sigma for summation, integration (simple integral and integral with circles), and pi for consecutive products. However, we plan to expand this capability to accommodate the interpretation of a variety of other symbols. We are also planning to experiment with other algorithms for the handwriting interpretation, as there are many more that we have not tried.

Currently, our system when running multiple slave interpreters is susceptible to occasional freezes due to large demands for the CPU time from one interpreter. Therefore, for smooth operation when running multiple interpreters, we are planning on using a thread extension to Tk. We expect that the use of such extension will prevent the occasional freezes caused by the hogging of the CPU time by some particular interpreters.

In our current system, the page-changing capability is limited to the presenter, and it is not possible for any of the clients to advance pages on their own screens. However, we plan to improve this and make page advancement possible for the client's side as well, while still maintaining the page number currently displayed on the presenter's screen. This will provide participants with more freedom to look back as their needs entail, while enabling them to return to the current page at any time.

Overall, our new whiteboard has turned out to be a unique display of the strengths of the Tcl/Tk language. It combines the language's easy embeddability of C modules, simple scripting syntax, and multiple interpreter capability into a useful networked digital whiteboard system. Although due to its short development period, it still lacks refinement in the ease of operation and GUI design, we believe that further development including the above enhancements, will make this software a convenient and valuable application in various fields of business and research.

References

[1] W. Confer and R. Chapman. Handwritten character recognition for cheap, 2002. 5

[2] S. Jaeger, S. Manke, and A. Waibel. Npen++: An online handwriting recognition system. In *in 7th International Workshop on Frontiers in Handwriting Recognition*, pages 249–260, 2000. 5

[3] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley Publishing Company, 1994. 2

[4] L. L. Peterson and B. S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, second edition, 2000. 1

[5] B. Welch, K. Jones, and J. Hobbs. *Practical programming in Tcl and Tk (4th ed.)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003. 1