Freehand Formula Entry System Version 0.3 User's Manual

Richard Zanibbi
Diagram Recognition Laboratory
Queen's University
Kingston, Ontario, CANADA

April 30, 2004

1 Introduction

The Freehand Formula Entry System (FFES) is an interpretive interface for entering mathematical notation using a mouse or data tablet, implemented in C++ and Tcl/Tk. This is the third version (0.3) of FFES. The first version was completed by Steve Smithies at the University of Otago New Zealand under the supervision of Kevin Novins, and with the assistance of James Arvo (California Institute of Technology) in 1999. A number of others have contributed to the code since that time.

Research publications about FFES include Steve's Master's Thesis and first conference publication about FFES [1, 2], a journal article [3] and a paper describing an experiment with the 'Align' operation [4]. Many of the alterations made in version 0.3 were in response to comments by participants in that experiment, and from comments made by attendees of the Graphics interface 2001 conference where that work was presented.

In the remainder of this section we list the people who have contributed to the current system, describe the applications used by FFES, describe the FFES subdirectories, and provide some brief notes on distributing and installing FFES. In subsequent sections we provide a quick introduction and reference for the different editing, file and display operations available in FFES, some pointers on where to start if you wish to alter or extend FFES, and finally contact information for sending us bugs, comments and suggestions.

1.1 Contributors

The people that have contributed code to this version of FFES include Steve Smithies, Kevin Novins, Richard Zanibbi, Arlis Rose, David Tausky, and Nick Willan.

1.2 Applications Used by FFES

FFES makes use of a number of other applications, which are distributed along with it. These are:

1. Caltech Interface Tools - a nearest neighbour symbol recognizer along with a set of tools for training the recognizer and using results in interpretive interfaces. CIT was developed at the California Institute of Technology by James Arvo and others (see CIT4.3/README.html).

2. DRACULAE - a set of TXL programs used to parse mathematical expressions described as a list of symbols with bounding boxes. DRACULAE is used to analyze the symbol structure (syntax) and semantics of an input expression. DRACULAE has its own set of documentation available in the DRACULAE_0.3/doc directory.

FFES also makes use of LaTeXand **dvi2bitmap, which are usually installed by default on Unix-based systems.**

1.3 Copying and Distribution

You may freely distribute FFES in accordance with the terms of the GNU General Public License (GPL). A copy of the license is in the LICENSE file in the FFES_0.3/ directory.

1.4 Subdirectories

Here is a summary of the FFES subdirectories:

bin/: holds the FFES executable ffes.

doc/: contains the FFES documentation that you are now reading.

examples/: example expressions, including some that demonstrate known bugs of the system.

images/: contains images used by the program, including thumbnail images used in the pop-up menu for symbol correction.

src/: C++ and Tcl/Tk source code, organized into:

ffes/: C++ source for the main ffes program. Include code to start the Tcl/Tk interface, and character recognizer built using the CIT library.

Tk_Interface:/ code for the Tcl/Tk interface.

temp/: holds intermediate data used by FFES, including undo files in the undo_files/ subdirectory.

The remaining subdirectories of ffes_0.3/ contain the applications used by FFES.

1.5 Installation

Please consult the INSTALL file in the parent directory (ffes_0.3).

2 FFES Operations: Quick Introduction and Reference

2.1 Quick Introduction: Using FFES for the First Time

When first using FFES, we recommend starting by loading some of the examples from the examples/ directory, and trying out the different operations available from the menus and editing modes. As another approach, you could try drawing '2 + 2' in Draw mode. Then use Correct mode to correct any symbol recognition errors (see Section 2.5). Then go back to Draw mode, and try using each of the operations under the Display menu (see Section 2.7). Try using Select mode to move and delete symbols (see Section 2.4). Finally, try the Align operation (see the next subsection). Note that you have Undo/Redo buttons to reverse or re-perform any editing operations you have used.

The remaining parts of this section are intended to provide a quick reference for the operations of FFES.

2.2 Mode-Independent Editing Operations

In this section we briefly describe the mode-independent editing operations of FFES. All of the operations described in this section may also be found under the Edit and Operations menus of FFES, or as a button in the interface (e.g. Align, Undo/Redo).

Editing modes (a,s,d): there are three modes in FFES: Correct, Select and Draw. These modes can be entered by pressing the corresponding button or by pressing a,s or d, respectively (see the following sections for details of the operations available in each mode).

Undo/Redo (Ctrl-z, Ctrl-r): Undo/redo an editing operation.

Align (Ctrl-a): This will resize and align symbols on the baselines detected by DRACULAE. You can use this to clean up your input, or to view recognition results. Note that the operation currently only works on individual symbols (e.g. for $\cos x$, the *c,o,s* and x will be evenly spaced along the baseline, not grouped together).

Edit options: (Ctrl-g): will bring up a dialogue allowing you to configure the appearance and behaviour of FFES. Note that currently the option values are not saved.

Copy text to clipboard (Ctrl-v): this will copy the current contents of the text panel in the lower right hand corner of FFES to the X clipboard. You will need to run Xclipboard to be able to view/reuse the copied data. Note that alternatively you can select and then middle-click to copy the contents of the text panel to another application.

Force symbol recognition (r): don't want to wait for the symbol recognizer to start up? Just press r to force the recognizer to return results. Note that the symbol recognizer timeout may be set using the Edit options command (Ctrl-g).

2.3 Draw Mode Operations

mouse button 1: draws a line ('stroke') on the canvas.

mouse button 3: if the mouse is over a symbol's bounding box, this pops up a list of symbol types returned by the recognizer, as well as a list of other available symbol types. If the mouse is not over a symbol's bounding box, this starts the stroke segmenting tool (see Correct Mode Operations).

2.4 Select Mode Operations

mouse button 1: If no symbol has been selected:

1. **clicking on a bounding box will select it, and you may move the symbol.**
2. **clicking off of a bounding box will start the corner of a selection box: drag the box over the symbols you wish to select. Symbols will highlight as they are selected.**

If a symbol or group of symbols has been selected:

1. **Clicking off the selected group deselects the symbols and starts the drag-select tool.**
2. **Clicking on a selected group of symbols:**
   (a) **Drag the symbols around the canvas to move them.**
   (b) **Deleting symbols: you may press the Delete button (shortcut: Ctrl-x), or drag the symbols below the bottom of the canvas. The mouse cursor will change to a skull and crossbones to indicate that letting go of the mouse button will delete the selected symbols.**

mouse button 3: if the mouse is over a symbol's bounding box, this pops up a list of symbol types returned by the recognizer, as well as a list of other available symbol types. If the mouse is not over a symbol's bounding box, this starts the stroke segmenting tool (see Correct Mode Operations).

2.5 Correct Mode Operations

mouse button 1: if the mouse is over a symbol's bounding box, this pops up a list of the symbol recognition results for the symbol, as well as the list of available symbol types. You may select Enter Name to enter a text label using the keyboard.

If the mouse is not over a symbol's bounding box, the stroke segmenting tool is started.

stroke segmenting tool: this tool allows the user to drag the mouse over a group of strokes that they wish to join into a single symbol. Selected strokes which belong to other symbols are removed from their original symbol and added to the new symbol created by joining the selected strokes. Individual strokes may be separated into a single symbol by selecting just the single stroke. Strokes are highlighted as they are selected.

mouse button 3: starts the stroke segmenting tool, regardless of whether the mouse is over a symbol's bounding box or not.

2.6 File Operations

Each of the following operations are also available under the File menu.

New (Ctrl-n): clear the canvas.

Open (Ctrl-o): open an .ffes file.

Save (Ctrl-s): save the current expression as an .ffes file containing all the current stroke and symbol information. Save to the current file if it has been named, and otherwise prompt the user to provide a name and location for the file (Save-as).

Save as (Ctrl-e): prompts for a name and location to save an .ffes file corresponding to the current stroke and symbol information.

Save Canvas as Postscript (Alt-p): Save the canvas image as a postscript file.

Save Symbols as .dat file (Alt-z): Save the labels, bounding boxes and symbol id's of the canvas symbols in a .dat format file.

Save .gif Image (Alt-g): save the bitmap feedback image.

Print (Alt-p): print the canvas (send postscript to a printer).

Exit (Ctrl-q): exit FFES.

2.7 Display Options

In addition to the display options in the options menu (Ctrl-g), we have added a number of options to control what information is displayed to the user.

Auto Update Bitmap: normally you'll want to leave this set to 'on'; if set to 'off' the bitmap generated from the recognition result will not be updated each time the symbol state changes.

Show/Hide Foreground (f): show/hide the user-drawn strokes.

Show/Hide Background (b): show/hide the symbol labels that appear behind the user-drawn strokes. Note: in Correct mode the symbol labels are always shown.

Show/Hide Bounding Boxes in Draw Mode (under Display menu): show/hide the symbol bounding box outlines in draw mode.

Show/Hide Bounding Boxes in Select Mode (under Display menu): show/hide the symbol bounding box outlines in select mode.

Show/Hide Bottom Panel (F1): show/hide the bitmap and text panels.

Show/Hide Text Panel (F4): show/hide the text-panel in the bottom-right corner of FFES.

Text Panel (F5-F9) : the text panel at the bottom right-hand corner of FFES shows the syntactic and semantic interpretation of the current symbol layout; the TEX, operator tree and intermediate outputs of DRACULAE may be seen by selecting the appropriate radio button at the bottom of the text panel, or by pressing F1-F5 (Ctrl-v can be used to copy the current text panel contents to the clipboard; see Section 2.2).

3 Altering and Extending FFES

In this section we provide some quick pointers for people who wish to extend or alter FFES.

event bindings(mouse, keyboard): Interface.tk and all the bind_* files in Tk_Interface specify the main mouse and keyboard behaviour for FFES. menu_button_appearances.tk and thumbnails.tk contain additional bindings (achieved using the bind command in Tcl/Tk).

character recognizer (CIT): the C++ code written by Steve Smithies to interface to CIT through Tcl is located in the src/ffes/ subdirectory. Additional Tcl/Tk code controlling how the symbol recognition results are obtained and requested is in Tk_Interface/symbol_recognizer.tk. Alternative character recognizers can be used by altering this code.

parser (DRACULAE): the files parse_symbols.tk, and edit_align_symbols.tk specify how DRACULAE is called to obtain TEX and alignment of symbols, respectively. You may change the arguments to DRACULAE (e.g. to change thresholds) as described in the DRACULAE documentation (see DRACULAE_0.3/doc/). You can also try out alternative parsers by replacing the calls to DRACULAE with calls to another system.

bitmap generation: currently a .gif bitmap is generated using dvi2bitmap by the function Update_Bitmap in Tk_Interface/parse_symbols.tk. If image rendering is slow when running FFES, check that dvi2bitmap is not having problems finding fonts. Consult the dvi2bitmap documentation for more details.

thumbnails: thumbnails are read at run-time from the images/thumbnails/ directory. If you examine the pop-up correction menu window, you will see that the subdirectories of images/thumbnails/ appear as headings, with the images of each subdirectory below the appropriate heading. The files Tk_Interface/thumbnails.tk and Tk_Interface/mode_modify_characters.tk are the most important files for the thumbnails and pop-up correction menu.

drawing routines: most drawing routines may be found in Tk_Interface/drawing_routines.tk.

background fonts: the code that produces the images showing symbol recognition results behind user drawn strokes are found in Tk_Interface/font_stroke.tk and Tk_Interface/drawing_routines The font files used are from CIT, in the CIT4.3.1/fonts/fdf directory.

4 Bugs, Comments and Suggestions

If you locate a bug while using FFES, please save the problem expression as an .ffes file and send it with a message describing the bug to zanibbi@cs.queensu.ca. If this is not possible or appropriate, simply send a message describing the bug.

If you have any comments or suggestions about FFES, we're interested in hearing them. Please send your comments and suggestions to zanibbi@cs.queensu.ca.

Acknowledgements

I wish to thank my supervisors Dorothea Blostein and James R. Cordy for their support. This research has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the National Science Foundation (USA), under Career Award CCR9876332.

References

[1] **S. Smithies. Freehand formula entry system. Master's thesis, University of Otago, Dunedin, New Zealand, May 1999.**

[2] **S. Smithies, K. Novins, and J. Arvo. A handwriting-based equation editor. In *Proc. Graphics Interface*, Kingston, Ontario, Canada, June 1999.**

[3] **S. Smithies, K. Novins, and J. Arvo. Equation entry and editing via handwriting and gesture recognition. *Behaviour and Information Technology*, 20(1):53–67, 2001.**

[4] **R. Zanibbi, K. Novins, J. Arvo, and K. Zanibbi. Aiding manipulation of handwritten mathematical expressions through style-preserving morphs. In *Proc. Graphics Interface*, pages 127–134, 2001.**