FrameMaker Palettes

Author: Dan Emory

Publication Date: 28 August, 1999

1 How Do You Create a Palette?

Any FrameMaker file can be turned into a palette by first saving it as a MIF file, and then editing the MIF to change the values of the following parameters (as they are defined on Windows platforms):

`DViewOnly`: change from No to Yes

`DViewOnlyWinBorders`: Change from Yes to No

`DViewOnlyMenuBar`: Change from Yes to No

`DViewOnlyPopup`: Change from Yes to No

`DviewOnlyWinPalette`: Change from No to Yes

I typically create palettes with a page size of 3.5" x 6.25", as shown in Figure 1.

The only way to get action out of a palette is by embedding a separate hypertext marker for each action provided.

If you create a **multi-page palette**, you must include next/previous page or goto-type hyperlink buttons on each page, because that's the only way you can change pages. Or, you could create a button that has a popup hyperlink in it that produces a popup menu, listing the possible destinations within the palette.

2 How Do You Change a Palette Once It's Created?

Keep the original (FrameMaker native binary format) file, which (by default) has all of the DView settings reversed from those that produce the palette behavior. Edit that file, again save it as MIF, and again change the DView settings as described in Section 1 to convert it to a palette.

3 Palette Behavior

With the DView settings described in Section 1, you open a palette in the same manner as an ordinary FrameMaker file. It will appear as a view-only document that is always in front, without the normal borders. It does not respond to (or interfere with) actions you take in the menu bar or dialog boxes (you dismiss the palette by clicking in the upper-left corner, just as you would dismiss any other built-in FrameMaker palette). You also move the palette in the same manner as a built-in palette, by simply clicking on the top border and dragging it to where you want it.

EMAIL: DANEMORY@PRIMENET.COM PHONE: 949-722-8971

4 What Can You Do With a Palette?

Anything you can do with hypertext links in an ordinary view-only FrameMaker document you can do in a palette. Using the `message` hypertext command, you can open other documents and launch other applications or API clients.

FM+SGML has a special API client called FmDispatcher. To use it for the purpose I will describe below, the read/write rules for your SGML import/export applications must include the statement:

```
#include isoall.rw
```

This statement invokes the built-in read/write rules for all the ISO PUBLIC SGML character sets, which include most special characters, as well as some multi-character combinations (e.g., fractions). Each such character in the ISO character sets is assigned an unique entity name.

Now, you can create a palette that allows you to insert special characters into your documents without having to look up the ANSI number or the FrameMaker escape sequence.

I've created a 3-page palette for that purpose (shown in Figure 1). The first page contains miscellaneous special characters (e.g., registered and trademark symbols, ellipsis, em-dash, en-dash, dagger, etc.) plus math symbols that require the symbol font. The second page contains the upper- and lowercase Greek letters, and the third page contains commonly used fractions. In my palette, there is an actual representation of each character. Embedded in the text line containing each character is a hypertext marker of the form:

```
message FmDispatcher insert entity entname
```

Where `entname` is the ISO PUBLIC entity name for that character.

So, to insert a special character, you simply put the text cursor at the insertion point in your document where you want the character to appear, and click on the desired character in the palette. Here's what happens:

1. The palette sends the `message` hypertext command to FmDispatcher.
2. FmDispatcher looks in the read/write rules for the currently specified SGML import/export application, finds the `#include isoall.rw` statement, looks up the entity name in the ISO read/write rules, and inserts at the insertion point what the read/write rule for that entity tells it to insert. In the case of special characters requiring the symbol font, the character is wrapped with a character tag that specifies the symbol font. In the case of fractions, a variable is inserted whose definition produces the fraction. When you export such a document to SGML, each such character or variable inserted from the palette will be converted to the corresponding ISO PUBLIC entity reference.

It turns out that, if you are using FM+SGML to produce an unstructured document, the palette works there as well. All you have to do first is to set the SGML application to one whose read/write rules have in it the `#include isoall.rw statement`. In this case, FM+SGML simply ignores the fact that the document is unstructured, and inserts the character without complaining about it.

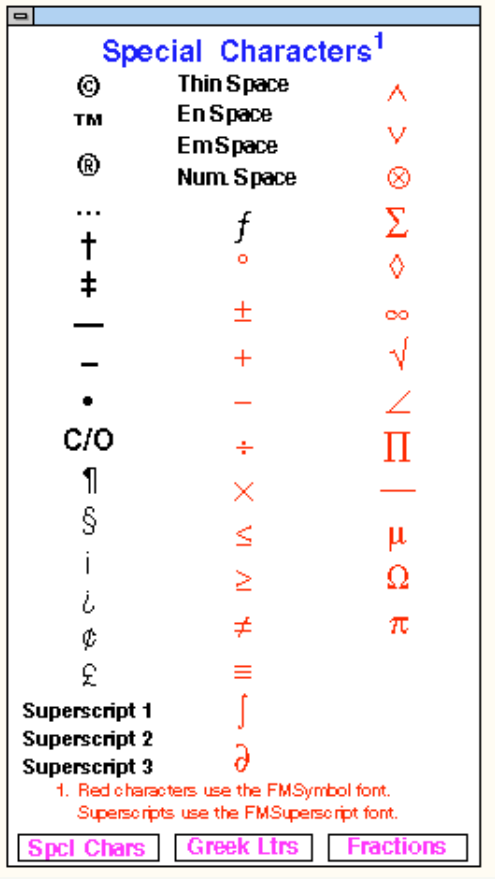


Figure 1. Special Character Palette for FM+SGML[1]

5 What are the Possibilities?

Suppose FrameMaker V6 has included in it something like FM+SGML's FmDispatcher API that responds to message hypertext commands that are directed at it. Suppose further that V6 has built into it a platform-independent, high-level, macro language, allowing each user to create his/her own macros, using that language. Each user could create a palette that lists/describes the macros he/she has created. Embedded in the palette line for each macro description would be a hypertext link of the form:

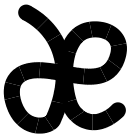
```
message FmDispatcher execute macro [macroname parameters]
```

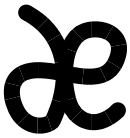
When you click on the macro description in the palette, FmDispatcher would look up `macroname` and execute it, using the specified `parameters`.

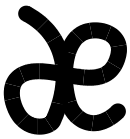
Other, even more exotic, functions could also be built into the V6 FmDispatcher to provide users with even greater customization possibilities, employing user-developed palettes and message hypertext commands to execute each such function. I believe this approach (combined in the case

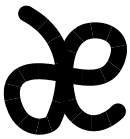
1. Those of you who have FM+SGML can obtain a copy of the palette shown in Figure 1 by emailing a request to me at danemory@primenet.com.

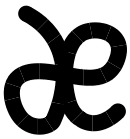
of FM+SGML with more powerful read/write rule functions) could possibly eliminate as much as 90% of conventional (C++) API development, thereby substantially reducing the cost of customized development. If, for V6, it were a choice between a new GUI and something like I'm describing here, I think everyone would choose the latter.

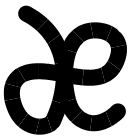


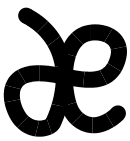


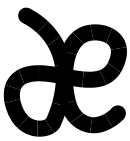












Dan Emory & Associates

Information Design Specialists

Dan Emory

▬

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

▬

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

▬

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

▬

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

▬

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

▬

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

▬

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

▬

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com