NSDISPLAYSIZES

By: Bill van Melle (vanMelle.pa@Xerox.com)

The NS font families all have screen fonts that display at approximately their nominal point size. This means that there is a closer congruence between their appearance on the display and on hardcopy than there is for, say, the Press fonts. Unfortunately, this means that the NS display fonts are "too small"—a size that is quite readable on hardcopy can be unomfortably small on the display. The module NSDISPLAYSIZES attempts to ameliorate this problem by fooling FONTCREATE into using bigger fonts for display without changing anyone's belief in the nominal size of the font.

Loading NSDISPLAYSIZES.LCOM modifies FONTCREATE's font file lookup procedure so that a request for NS display font of size *n* actually reads the font file for size *n*+2. For example, (FONTCREATE 'MODERN 10) will actually read the display font file belonging to Modern 12, but FONTCREATE will still believe the resulting font is Modern 10. Font sizes greater than 12 are not affected, on the grounds that those fonts are already big enough to read, and not all fonts are available in size *n*+2 for large *n*; hence, for example, Classic 12 and Classic 14 will end up using the same actual font for display. Also, since Terminal 14 does not yet (as of this printing) exist, Terminal 12 remains Terminal 12.

A font is considered an NS font if its name is a member of the list NSFONTFAMILIES, whose initial value is (CLASSIC MODERN TERMINAL OPTIMA TITAN).

Loading the module clears the internal font cache of all NS display fonts, so that subsequent calls to FONTCREATE will not erroneously return a font cached earlier under the default lookup procedure. Of course, if someone has already set some font variable to (FONTCREATE 'MODERN 10), that font descriptor will not be affected.

Note that this module has no effect on hardcopy—a font is always printedat the size you named it. And you can still use TEdit's Hardcopy display mode to see how a piece of text will be formatted on the printer.

If the VIRTUALKEYBOARDS module is present, then loading NSDISPLAYSIZES automatically edits its keyboard specifications so that it continues to use Classic 12 in its keyboard displays. Without this fix, the keyboard display routines will try to create Classic 12, which NSDISPLAYSIZES coerces to Classic 14, and as a result the keyboards will be poorly displayed and lack many characters (since Classic 14 is not as complete as 12). If you load VIRTUALKEYBOARDS *after* NSDISPLAYSIZES, you should call (VKBD.FIX.FONT) yourself to make the change.

Note that other modules can have similar problems if they have hardwired into them a specific size of NS font as being appropriate for their display configuration. For such modules to operate correctly in the presence of NSDISPLAYSIZES, they might want to be aware of the function used to coerce sizes:

`(NSDISPLAYSIZE` *FAMILY SIZE FACE EXTENSION*`)` [Function]

Returns a font size that `(FONTCREATE` *FAMILY SIZE FACE*`)` will use instead of *SIZE*. *EXTENSION* must be a member of DISPLAYFONTEXTENSIONS in order that we know we are doing this for the display. Follows the rules described above. For example, `(NSDISPLAYSIZE 'MODERN 12 'MRR 'DISPLAYFONT)` returns 14. `(NSDISPLAYSIZE 'GACHA 12 'MRR 'DISPLAYFONT)` returns 12.

In the simplest case a module could just test for the existence of NSDISPLAYSIZE and choose one size or another. For example,

```
(SETQ MYFONT (FONTCREATE 'MODERN (if (GETD 'NSDISPLAYSIZE)
                                     then 10
                                    else 12)))
```