TERMiTE[1]

by Lloyd Milligan[2], Tom Ackerman[3], Edward J. McIntosh[4]

You have a first-rate M application that has worked admirably for the past number of years. Suddenly your client is unhappy. You probe cautiously to discover the reason for his or her change of mood. Yes, everything is working as it should. No, there have not been any problems with the application. Yes, the job is getting done as effectively as ever. Finally, a confession emerges. Your client has seen another application, similar to yours. It has most of the functionality and seems to work. But, wow does it look nice! Instead of plain screens filled with text labels and data fields, the newly discovered product is three-dimensional and ablaze with color. Its screens boast pulldown menus, popup windows, pushbutton controls, and all those wonderfully stimulating inventions that are commonly referred to as the graphical user interface[5].

When asked what the M community should do to move forward, John Dvorak, who served as keynote speaker for the 24th Annual Meeting of MTA in Chicago[6], named the Graphical User Interface as a top priority. This suggestion is hardly news to M developers. Many programmers are currently struggling to convert host-based "legacy" systems to client/server computing, featuring workstations and the graphical user interface. M is not the only technology undergoing this transition. A glance at any trade magazine confirms the universal nature of similar modernization initiatives.

The reader may wonder what these remarks have to do with terminal emulation. After all, we know how to put a graphical user interface on an M application. A variety of tools are available including the M Windows API, Visual Basic, Delphi, and so forth. With each of these tools or approaches, a significant amount of programming is needed at both ends. Few applications are written in such a way that a conventionally designed graphic interface can be added to the front end without modifying the application internally. Furthermore, programming the interface itself using one of the major object oriented technologies is not an insignificant chore.

The unique appeal of TERMiTE is the ease with which a graphic interface can be designed for a legacy application that currently uses "dumb terminal" screens for input. TERMiTE itself resides on a PC and does not require M to be installed on the PC. It interfaces to the M environment in the same way that terminals interface, through a serial or network connection. While TERMiTE is not the only product to address the host-application-to-graphic-interface conversion problem, it offers the dual advantage of application independence and minimal programming.

TERMiTE is the leading terminal emulator in the PICK language world. "Never heard of PICK you say." Well now you know how most people feel about M. Interestingly, PICK and M share some common features, not the least of which is that both are "niche" languages. PICK, like M, has thousands of applications running on hosts worldwide, many performing critical database functions. Facing the same pressure to modernize the user interface, PICK developers conceived an unusually simple solution.

Instead of undertaking a major redesign of their legacy systems, these developers opted to place the graphical user interface on the workstation or PC and to conserve as much legacy code as possible. In this way the same host application could serve concurrently the existing dumb terminal base and the graphic workstation during the transition. TERMiTE is the tool that made this strategy possible.

For organizations committed to rewriting their host-based applications for client/server, TERMiTE offers a uniquely attractive migration strategy. Using this product, the inventory of hostbased applications can be converted in an incremental fashion while with a modest investment, a graphical user interface can be offered for both converted and legacy applications on all PCs as they replace terminals.

Dumb terminals remain in widespread use throughout the industry. For high-speed large-scale data entry applications where operators infrequently look at the screen, dumb terminals are as effective as PCs and cost less, thereby justifying the "dual" approach to conversion made possible by the enhanced terminal emulator.

To appreciate the full range of TERMiTE's capabilities, it is necessary to acquire a copy and try out its various features[7]. However, we will attempt to convey an idea of what can be done with little or no programming using TERMiTE. First, suppose you have an application that uses ANSI line drawing to display borders on your data entry screen. In a well-designed application, the escape sequences are contained in tables, one entry for each type of display terminal. TERMiTE can emulate a variety of terminals including the popular DEC VT series. If you turn on sculpturing in TERMiTE, ANSI line and box drawing sequences produce a three-dimensional (sculptured) effect instead of the usual thin horizontal or vertical lines. Immediately, with no modifications to host code, the screen has a more interesting and satisfying appearance.

A good host application also provides hooks to execute code at various levels. For example, VA ScreenMan has pre- and post-actions associated with forms, pages, blocks, and fields. Judicious use of such connection points can minimize the programming required to modernize an application. For another example, let's say you use the PF keys to perform generic functions such as save screen, exit, help, next screen, prev screen, and so forth. Using TERMiTE you can create controls (buttons) to perform these functions. Furthermore, you can request that TERMiTE substitute an alternative message for an event, sending the host the same PF key sequence that the application expects to receive from the keyboard. It is remarkable enough that a useful subset of graphical functions can be attached to an application without changing the application internally. However, an even more impressive range of functionality can be had at the cost of adding conditional code to use TERMiTE's advanced features.

In addition to screen manipulation functions, TERMiTE's Applications Interface Facility includes full support for ODE. Thus the host can initiate a conversation with any DDE server application, exchanging data transparently. This capability enables an M program to control almost any Windows application without involving the user. Finally, TERMiTE has its own BASIC-like macro language through which the programmer can expand its functionality as desired.

Writing escape sequences directly to the device (or emulator) can be tedious and errorprone. Therefore, following the example set by PICK programmers, we have coded a small library of M subroutines and extrinsic functions to simplify communications with TERMiTE. For example, to create a "Help" button at row 19, column 72, of size 2 rows high by 8 columns wide and using default visible, enabled and font attributes, the following code can be used:

```
D TBUTTON^PIXEL(19, 72, 2, 8, "B1", "Help")
```

where "B1" is the name (control i.d.) assigned to the "Help" button. When the application must read an explicit response from TERMiTE (such as whether the user pressed "yes" or "no" in a message box), the program first turns echo off (using either TERMiTE or host device control), then queries TERMiTE using, for example:

```
S RESP=$$GETVALUE^PIXEL9 (TO)
```

where the parameter TO is a time out value. These two brief examples illustrate the general flavor of communication with TERMiTE via M library routines.

To be fair, it should be noted that TERMiTE does not express the full power and flexibility of a programming language such as C++ or Visual Basic. Its principle virtue, in addition to being a good terminal emulator, is the ease with which it can be incorporated into an existing host-based application to produce an immediate and satisfying improvement in the user interface. The developer is free to redesign archaic components of the host system without having to endure the time pressure that tends to prevail when no intermediate or transitional solution is feasible. **M**

NOTES

[1]Billed as "the world's most advanced terminal emulator," TERMiTE is a product of Pixel Innovations Ltd. (Pixel USA), 1316 Dunwoody Village Parkway, Suite 200, Atlanta, Georgia 30338 Telephone: 770-512-7417, FAX 770-512-7745. Pixel's world wide web home page may be found at http://www.pixel.co.uk/pixel/

[2]Sea Island Systems, Inc., PO Box 655, Isle of Palms, SC 29451, email wlloyd@ix.netcom.com

[3]M Systems Plus, Inc., 9265 Waits Ferry Crossing, Duluth, GA 30136

[4]American Custom Software, 2484 Brafferton Way, Atlanta, GA 30360

[5]A colleague, Bain Henderson, points out the significance of the psychological truism that humans are "stimulus-seeking organisms." Given equivalent functionality in things of varying stimulus potency, humans are most likely to choose the more stimulating alternative.

[6] John M. Dvorak, "Dvorak on M: Part II", *M Computing* (3) 15, November/December, 1995.

[7]A free 14-day evaluation copy of TERMiTE is available.