In 1996, I retired (for the second time) and, with time on my hands, decided to create a computer program to identify mushrooms to the species level. Over the years, I had had some experience designing specialized database programs and was interested in the possibilities of using the power of computers to identify wild mushrooms.

My original plan: obtain a digital data set of mushroom species and their characteristics from the U.S. government, or from university researchers; create a way for a user to enter similar data into a computer, and create a search engine to compare the two datasets. Estimated time to completion: two or three months, tops.

Well, that pipe dream lasted about as long as it took to make a few calls and search the internet. There is no such data set, at least not that I could find. Being brave and more than little stupid, I decided I could create this database. I mean, how hard could it be? Right? I won't bore the readers with the effort it took to research and record identification characteristics for 1,000 species of wild mushrooms, the arbitrary number I had selected. However, I will say that it took months.

The next step was to create the data entry program. The plan was for the user to point and click to describe the unknown mushroom to the program. I had not done any serious programming for the past 12 years, and Windows had passed me by. So, not only did I have to create a serious data set, but I also had to play catch-up on Windows programming. Since I have used LMI Forth packages from the early days, I decided to use LMI's WinForth.

WinForth had just about everything I needed, including the hooks into the operating system. But some of the commands to create check-boxes and other items in dialog-boxes were difficult to use and required considerable trial and error. The result was less than satisfactory. I decided to abandon the use of WinForth to create the dialog boxes and, instead, use Borland's Resource Workshop, which has drag-and-drop and other functions specifically intended to make it easy to create dialog windows and menus.

After the dialogs and menus are created and stored in the .exe file as a static resource, they can be called as needed from WinForth. E. g., show-about " about" ['1 about-dialog loaddlg drop ; In the example, " about"is the name of the static resource in the .exe file, " about-dialog" is the dialog handler which sets up lists and then, when the dialog is exited, recovers user actions. Loaddlg is the WinForth command to load the dialog.

This combination of WinForth and Borland Resource Workshop made for a fast and powerful development system.

A great-looking dialog window with numerous checkboxes, bitmapped graphics buttons, and lists can be created in a matter of a couple of minutes. And this was invaluable, as the program eventually consisted of over 200 bitmaps, 30 menus, and 65 dialogs. Some of the dialogs had more than 20 buttons, checkboxes, lists, and other items.

Because I elected to use some specialized Borland items within the dialogs, it was necessary to load the Borland library at the start of the program: loadlib DSO " bwcc.dl1" asciiz loadlibrary equ borlib ;

The command loadlibrary is a WinForth command to call the Windows API to load the library. The variable borlib is used to hold the return so that the library can be released when the program exits, i.e., borlib free library. The remainder of the programming was pretty straight-forward, except for the search routines. Because of the ambiguity of many descriptions (e.g., is the cap brownish red, reddish brown, or perhaps rust brown?) a form of fuzzy logic was used in the search routines. For each attribute, an "importance" and a "nearness" value was assigned. This allowed the program to find the correct mushroom even when some of the data entered were not identical to the data stored in the database.

If I were designing a Windows-based Forth, I would omit all commands to create buttons and other dialog items, and concentrate on building in commands to integrate third-party software and to make it easy to use the Windows API functions. The availability of programs like Borland's Resource Workshop and Microsoft's Help Workshop, which can be used by any programming language, make it unnecessary to reinvent the wheel. The combination of these programs makes it a snap to create sophisticated user interactions.

I learned a great deal during the endeavor, and it was well worth the effort, even if it turns out a complete commercial failure. The fun was in the doing.

I became involved with Forth when I wrote a weather observation database program in BASIC for an early Apple. It was so slow that it was unusable, and Forth was the only other language available. I ported the program to Forth, and it ran great. The program was ported to LMI Forth shortly after the IBM PC first appeared.

The mushroom identification program is my first commercial venture and, while it does not have glitzy graphics, it works, and it is the first wild mushroom identification program on the market.