GUIBuilder: The Easy Way To Go GUI

By Jim Douglas

Graphical user interfaces (GUIs) have revolutionized the way end users interact with software. Instead of typing in memorized commands, they can now simply point and click their way through the most complex operations. The result has made computing much easier for average users but created a steep learning curve for developers.

GUI programs are structurally different from character programs, with the screen transformed into a graphical window and data fields replaced with graphical controls such as push buttons, check boxes, and radio buttons. Then there is the event loop, the heart of the event-driven GUI program, that cycles as it waits for the user to take some action--push a button or click on a menu--and then responds. And to make it even more complicated, windows and graphical controls are identified with arbitrary numbers instead of easy-to-remember names.

But a new tool in Visual PRO/5® 2.0 is simplifying the creation of event-driven GUI programs for developers at any experience level. GUIBuilder™ is a complete visual programming environment that lets developers focus on what's really important--designing the overall flow of the application and creating the business rules--by automatically handling all the housekeeping details that go into an event-driven GUI program. GUIBuilder even includes a series of defined functions that lets you work with graphical controls using meaningful names rather than arbitrary numbers. For developers who may have seen GUI as intimidating and overly complex, GUIBuilder makes GUI programming easy and understandable.

The Visual Programming Environment

The first step in developing a program with GUIBuilder doesn't actually start with GUIBuilder but in ResBuilder™, the new, visual resource builder in Visual PRO/5 2.0. Using ResBuilder, you create a window--a type of GUI form--and then place graphical controls in the window. ResBuilder stores the structure of the window and controls in a *resource file* (see "Randolph Wrestles with ResBuilder" in *The BASIS Advantage* at *www.basis.com/advantage/mag-v2n1/randolph.html*). Then you can open the resource file in GUIBuilder and begin selecting forms, windows, controls, and events from lists, and type in *event handlers*--blocks of Visual PRO/5 code

that tell GUIBuilder how to respond to selected events. When you've finished creating event handler subroutines, GUIBuilder generates a complete and extensively commented GUI program.

The visual orientation of GUIBuilder eliminates the possibility of errors in identifying forms, windows, controls, or events by allowing you to pick only legal values from lists. To ensure that the program remains up to date with resource file changes, a detailed cross-check is done every time you load the program into GUIBuilder and when you return from editing the resource file in GUIBuilder.

GUIBuilder can't automatically convert all your character-based programs from character to GUI, but it does help you incorporate your existing character-based code into your new GUI programs. The GUIBuilder interface has two windows, an *Edit Window* for typing in blocks of code and a *View Window*, where you can load secondary text files or programs. When a tokenized Visual PRO/5 program is loaded into the View Window, GUIBuilder converts all line number references in the program to label references and then loads the program as formatted text. You can then easily cut selected blocks of code from your legacy program and paste them into the new GUI program, toggling back and forth between the Edit Window and the View Window.

The instant error checking available with the traditional BBx® console is not lost. GUIBuilder offers a menu option and corresponding tool button that can be used to check syntax at any time. This feature checks the syntax of the current block of code and returns either a "No errors found" message or a list of errors. When an error from the list is selected, the cursor is placed in an edit window with the incorrect line of code highlighted. Detailed syntax checking and reporting are also done when a completed program is built. Another option can be used to check for and report syntax errors automatically when saving each block of code to the work file.

In addition to the features described above, GUIBuilder lets you

- Create or edit a resource file in ResBuilder, while keeping your GUIBuilder work file in sync with resource file changes.
- Work with an optional data dictionary in DDBuilder™ (see "DDBuilder: New GUI Data Dictionary Editor," in *The BASIS Advantage* at *www.basis.com/advantage/mag-v1n1/ddbuilder.html*).
- Open a separate Visual PRO/5 console so you can try out tricky syntax interactively.

Program Framework

With GUIBuilder, the parts you don't see--the framework and the associated code-generation logic--are just as important as the parts you do see. In GUIBuilder, the code blocks that you create are

inserted into a standard framework that takes care of the housekeeping details important to any event-driven GUI program. The framework

- Opens and closes the SYSGUI device.
- Loads all the resources (forms, windows, and controls) from a resource file and assigns a unique context number to each window.
- Builds the event loop that checks the event queue and invokes event handler routines as necessary.
- Creates string templates so you can work with control names rather than numbers.
- Generates standard escape and error handlers.
- Pulls all of the pieces together to generate a complete program.

GUIBuilder generates a complete, working program even before you add your own code blocks. You can sit down alone or with a customer and design the overall look of the program using ResBuilder and then immediately generate a complete prototype program that demonstrates the application's look and feel. You can even leave the generated program at the customer's site. When combined with the resource file and a single called program, **`_qres`**, which is required at run time, the generated program is completely independent of the GUIBuilder environment. If you're curious about the overall program structure, programs generated by GUIBuilder are fully commented and readable.

GUI Building Blocks

A program created using GUIBuilder is made up of a standard framework, plus individual code blocks. The code blocks fall into one of the following categories:

- **Initialization.** This type of code block includes everything that you want to do at the beginning of the program, immediately before the event loop. Typically, you would open data files, define record templates, and initialize global variables here.

- **Event handlers.** The event handlers are individual subroutines invoked as required by the event loop. As a programmer, you don't have to be concerned with calling the event handler routines. You just have to decide what you want to do when a given event--such as a specific button being pushed--occurs.

- **End of Job.** This includes everything that you want to do at the end of the program, immediately after the event loop terminates. Typically, you would close data files and do whatever other cleanup might be necessary.

There's one other type of code block, known as Subroutines or

Functions, that is used for common subroutines, defined functions, or nonexecuting code like the TABLE, IOLIST, or DATA statements. Code defined in these blocks isn't managed automatically by the framework, so it's up to you to decide how it will be used.

There are a few more interesting features associated with creating event handler routines. When you create a new event handler, the code block doesn't start out completely blank. It's initialized with enough code to get you started--sometimes with as little a comment line as **`REM Button was pushed`**. For more complex events, the block is initialized with a series of comment lines and other code to let you know what control variables are available and the various options associated with the control. When you create a new event handler, GUIBuilder also checks the event mask for the selected window, warning you if the event mask needs to be changed in order to make the event visible in the generated program.

Data Management Functions

Programming languages got much easier when programmers were able to use variable names like **`product_id$`** and **`customer_name$`** rather than **`p1$`** and **`n2$`**. In the same way, GUIBuilder lets you work with controls using the names defined in ResBuilder rather than their control IDs, making code that much more readable. You don't need to worry about context numbers and control numbers directly--they're handled automatically by the data management functions.

If you've managed to avoid GUI programming so far, now is the time to give it a try. GUIBuilder's visual programming environment makes GUI program development intuitive, and the thorough help system makes it easy to get started. If you already have some experience in writing event-driven GUI programs using Visual PRO/5 1.0, you'll be impressed with how quick and easy the process has become with GUIBuilder. To try out GUIBuilder for yourself; you'll need only to upgrade to Visual PRO/5 2.0. This new version is available for Windows 3.1, 3.11, 95, and 98, as well as Windows NT 3.51 and 4.0.