Subject: Re: Using multiple top level bases created with GUIBuilder
Posted by MKatz843 on Mon, 24 Mar 2003 23:56:51 GMT
View Forum Message <> Reply to Message

There are a number of ways to do this.
One way is to create a place for each of your top level bases to store the ID of the other top-level base. That said, you have several options.

Imagine that you create both of the top-level bases from a single procedure. Within that procedure the two base IDs are known.

Method 1) Use pointers.

base_A_ID_ptr = ptr_new(0L) ;---- store a long integer.
base_B_ID_ptr = ptr_new(0L) ;---- store a long integer.

Now modify each of your top-level-base creation programs to accept a new keyword that will be this pointer. When you create base A, you don't yet know what base B's ID will be, but you can give base_A the pointer to where that ID will be stored.

base_A_ID = create_base_A(something, something,
base_B_ID_ptr=base_B_ID_ptr)
base_B_ID = create_base_B(something, something,
base_B_ID_ptr=base_A_ID_ptr)

At this point, the pointers are empty. Now, by setting the pointers, we give bases A and B knowledge of the other's ID.

*base_A_ID_ptr = base_A_ID
*base_B_ID_ptr = base_B_ID

The only caveat, is that you have to store the pointer-to-the-ID somewhere within each base so that it's accessible from within the event handler routine. It's convenient to store that information in a UVALUE of the buttons you're clicking or some easy to get to base or another, and then retrieve it with a widget_control, ID, GET_UVALUE=base_ID_pointer.

Method 2) Use SET_VALUE

Another way to pass information into an existing widget is with SET_VALUE. You have to again create a place to store the ID once you know it, and you have to make a SET_VALUE routine for the widget. When the base receives a SET_VALUE widget_control command, it takes the value and stores it somewhere where it can be retrieved by the event handler routine.

So you would do something like
base_A_ID = create_base_A(something, something)
base_B_ID = create_base_B(something, something)

widget_control, base_A_ID, SET_VALUE=base_B_ID
widget_control, base_B_ID, SET_VALUE=base_A_ID

Method 3) If widget A needs to know the ID of B but not the other way around, then you can create B first, then give A a keyword containing base B's ID.

base_B_ID = create_base_B(something, something)
base_A_ID = create_base_A(something, something, base_B_ID=base_B_ID)

Within base A you'll again have to find a place to store the ID for use during the event handler. Another place to store it is in the UVALUE of the buttons.

```
button = widget_button(somebase, VALUE='Show', $
  UVALUE={name:'Show', ID:base_B_ID} )
```

Here the UVALUE is a structure and the ID of the other base is stored in one of the structure fields.

Method 4) Use Objects
This is very similar to the pointer method, above. You create an object or objects that will contain the IDs. Pass them to the bases upon creation, then set the object's values properly.
Disclaimer: While objects are certainly the most elegant way to solve this (or most any) problem, they usually do require a bit more code, and familiarity with object programming in IDL (which is great). Failure to mention that there exists an object-oriented solution to such a question could provoke a kind-hearted, verbal tongue lashing from the list gurus.

I hope one of these ways will help. Any way you slice it, I think you'll have to make some modifications to the pre-fab code generated by the gui builder.

Best of luck,
M. Katz