Subject: Re: widgets and objects
Posted by savoie on Thu, 30 Oct 1997 08:00:00 GMT
View Forum Message <> Reply to Message

NOTE: what follows is valid for idl 4 (I don't have 5 yet, so I can't tell you if there's a better way to do this with pointers, I'd imagine so)

> I am trying to figure out widgets and I don't know if what i am trying
> to do is even possible - maybe it's just the wrong way to go about it.
> I have an object, during the initialization of that object I have a
> widget come up that has some checkboxes and an OK button for the use to
> select some choices during initialization.

You want to launch a widget, and it's event handler before initialization.

> Where i am encountering problems is when i call xmanager is still
> continues with initialization. Do i even want to call xmanager? And
> how do i pass back the choices that the user chose?

Sounds like you might want a /modal widget (blocking), but if you're only launching one small widget, it shouldn't matter.

Code before initialize
code that launches widget
code that initializes

> There doesn't seem to be an easy way to pass variables around with
> widget events. Is the only way to do this is by using UVALUE?

Matthew, you're very perceptive, there are no easy ways to pass variables around in widgets. There are a few tricks with UVALUE, but for what you want to do, it might be easiest to do something like this.

```
 ..................................................................... ..........
 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;
;; make the event handler for the launched widget.

PRO example_event, event

  ;; get the data handle.
  widget_control, event.top, get_uval=sdh

  ;; get the uval from the event.id
```

```
  widget_control,  event.id, get_uval=uv

  CASE uv OF
    'b1' : BEGIN
      message_in_here =  'Button One was pressed: plan accordingly'
    END
    'b2' : BEGIN
      message_in_here =  'Button Two was pressed: give up. '
    END
    'ok' : BEGIN
      ;; leave this mess
      widget_control, event.top, /destroy
    END
    ELSE: message_in_here, 'something broke'
  ENDCASE

  ;; if the widget still exists, store the data in the sdh(handle).
  IF widget_info(event.top, /valid) then $
   handle_value, sdh, message_in_here, /set
END


 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;;;;;
pro example
;;; example on how to pass variables to widges
;;;
 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;;;;;

  ;;code to set up widgets
  tlb =  widget_base(/column)
  but =  widget_button(tlb, value='BUTTON 1', uval='b1')
  but2 =  widget_button(tlb, value='BUTTON 2', uval='b2')
  ok = widget_button(tlb, value='OK', uval='ok')

  widget_control, tlb, /realize

  ;; introduce the varaible into scope here
  secret_data_handle =  handle_create()

  ;; set it to the uval of the top level base (tlb)
  widget_control, tlb, set_uvalue=secret_data_handle


  ;; use the /Modal keyword to keep from proceeding in this routine.
  xmanager, 'example', tlb, event_handler='example_event', $
   /modal

  ;; and here the scope is still valid here.
```

```
   handle_value, secret_data_handle, what_was_there

   print, 'the button last pushed before exiting gave this string:'
   print, what_was_there

END ;; example
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;
```

> It looks as if you have to set UVALUE to some variable (structure,
> vari, or otherwise) and you can then retrieve and set it during the
> event procedure. But then how can i retrieve it later? When the
> widget comes up the initialization function finishes. Then the
> variables that hold the choices go out of scope and are inaccessible
> to any other procedure. Do i have this all wrong? -matt

I think the above example explains this pretty well.

Hope this helps.

Mattie

--
Matthew H. Savoie Systems Technology Associates
Systems Analyst Supporting NOAA Profiler Network
ph. 303.497.6642 Demonstration Division/FSL/ERL
mailto: savoie@fsl.noaa.gov <URL:http://www-dd.fsl.noaa.gov/online.html>

Subject: Re: widgets and objects
Posted by davidf on Thu, 30 Oct 1997 08:00:00 GMT

View Forum Message <> Reply to Message

Matthew Hanson (matt@ktaadn.com) writes:

> I am trying to figure out widgets and I don't know if what i am trying
> to do is even possible - maybe it's just the wrong way to go about it.
> I have an object, during the initialization of that object I have a
> widget come up that has some checkboxes and an OK button for the use to
> select some choices during initialization. The choices available are
> dependent on data files read in during initialization and affects what
> happens next in initialization (data is being read in and the way it is
> organized is based on the users choices).
> Where i am encountering problems is when i call xmanager is still
> continues with initialization. Do i even want to call xmanager? And
> how do i pass back the choices that the user chose? I have it set up to
> call an event when the OK button is pressed. If the choices are valid

> in the checkboxes it closes the widget and returns the choices to the
> initialization. There doesn't seem to be an easy way to pass variables
> around with widget events. Is the only way to do this is by using
> UVALUE? It looks as if you have to set UVALUE to some variable
> (structure, vari, or otherwise) and you can then retrieve and set it
> during the event procedure. But then how can i retrieve it later? When
> the widget comes up the initialization function finishes. Then the
> variables that hold the choices go out of scope and are inaccessible to
> any other procedure.
> Do i have this all wrong?

I don't think you have it all wrong. I think you have explored this enough to understand the problems, you just haven't discovered the solution yet.

What you need is a modal or blocking widget that can collect information from the user and return it to the program that called it. This kind of widget is sometimes called a "pop-up dialog" or "form" widget and it is usually written as a function that returns the "form" information as the function result.

Frankly, most people who write these kinds of widgets use common blocks to communicate between program modules. But because common blocks always make widget programs less useful, I don't like to use them. In this case, I prefer to use a pointer to store the information collected by the pop-up dialog. Since pointers are global in scope, they are not destroyed when the modal widget is destroyed.

Modal widgets essentially block when they get to the XManager call in the widget definition module. So any code below the XManager call will get executed only after the widget is destroyed. This is where you need to collect the information that the program's event handler stored in the global pointer location and return to the caller of the program.

A good example of a widget of this kind is GetImage from my web page.

This, by the way, is just one of many IDL programming techniques that will be taught at my IDL Programming Techniques course Nov 18-21. There are just a few seats left. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/