Subject: How to cleanup an object with a non-modal widget method
Posted by wlandsman on Mon, 17 Mar 2014 21:24:31 GMT
View Forum Message <> Reply to Message

I have an object with a non-modal widget display method. While it is usually used in communication with other objects, I also want to have a simple wrapper to display the widget.

```
pro displaymap

oAstroMap = obj_new('AstroMap')   ;Create the object
oAstroMap.widgetdisplay          ;Create the display widget

return
end
```

While this works, it doesn't clean up after itself, leaving the heap area full. I can't put an OBJ_DESTROY statement before the RETURN because the non-modal widget doesn't wait for the widget to be destroyed, and executes the OBJ_DESTROY right away.

I suppose what I want is for the widget to destroy the underlying object when one presses the Quit button. But my experiments with "OBJ_DESTROY, self " have not been successful. Thanks, --Wayne

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by Matthew Argall on Mon, 17 Mar 2014 22:02:00 GMT
View Forum Message <> Reply to Message

A simple solution would be to turn "displaymap" into a function so that you can return your oAstroMap object reference. Then, when you are done with it, you can destroy it.

Maybe more to what you are looking for, check out the TLB_KILL_REQUEST_EVENTS and KILL_NOTIFY keyword to the Widget_Base function. You can have the callback routines kill the object when the top level base is destroyed.

I have another, more complicated solution if neither of these work for you... Is AstroMap object based on function/object graphics? Or is it made from the old widgets?

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by David Fanning on Mon, 17 Mar 2014 22:15:12 GMT
View Forum Message <> Reply to Message

wlandsman writes:

> I have an object with a non-modal widget display method. While it is usually used in

communication with other objects, I also want to have a simple wrapper to display the widget.

>
> pro displaymap
>
> oAstroMap = obj_new('AstroMap') ;Create the object
> oAstroMap.widgetdisplay ;Create the display widget
>
> return
> end
>
> While this works, it doesn't clean up after itself, leaving the heap area full. I can't put an OBJ_DESTROY statement before the RETURN because the non-modal widget doesn't wait for the widget to be destroyed, and executes the OBJ_DESTROY right away.
>
> I suppose what I want is for the widget to destroy the underlying object when one presses the Quit button. But my experiments with "OBJ_DESTROY, self " have not been successful.

I would write DisplayMap in such a way that you can get the object reference back from it (write it as a function, or return the object in a keyword). Then, I would just make sure I added the object reference to the info structure of the widget program that calls it, and destroy along with the other pointers and objects I was cleaning up in the widget cleanup routine.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

---

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by wlandsman on Wed, 19 Mar 2014 00:58:36 GMT
View Forum Message <> Reply to Message

Mathew, David -- thanks for the suggestions. Sorry if this post appears twice -- Google groups is driving me crazy.

On Monday, March 17, 2014 6:02:00 PM UTC-4, Matthew Argall wrote:
> A simple solution would be to turn "displaymap" into a function so that you can return your oAstroMap object reference. Then, when you are done with it, you can destroy it.

I am trying to write a wrapper for non-IDL users. I want them to be able to type "displaymap" at

the IDL prompt and view and manipulate the widget, without them needing to perform any additional cleanup after they click the widget QUIT button.
>
> Maybe more to what you are looking for, check out the TLB_KILL_REQUEST_EVENTS and KILL_NOTIFY keyword to the Widget_Base function. You can have the callback routines kill the object when the top level base is destroyed.

I'll look more closely at these but it still looks like the object has to destroy itself (and not just the widget), and I haven't had much luck with OBJ_DESTROY, self. Perhaps I need to also write the wrapper as an object.
>
> I have another, more complicated solution if neither of these work for you... Is AstroMap object based on function/object graphics? Or is it made from the old widgets?

No, it uses "classic" widgets with Coyote graphics. --Wayne

---

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by David Fanning on Wed, 19 Mar 2014 03:03:59 GMT
View Forum Message <> Reply to Message

wlandsman writes:

>
> Mathew, David -- thanks for the suggestions. Sorry if this post appears twice -- Google groups is driving me crazy.
>
> On Monday, March 17, 2014 6:02:00 PM UTC-4, Matthew Argall wrote:
>> A simple solution would be to turn "displaymap" into a function so that you can return your oAstroMap object reference. Then, when you are done with it, you can destroy it.
>
> I am trying to write a wrapper for non-IDL users. I want them to be able to type "displaymap" at the IDL prompt and view and manipulate the widget, without them needing to perform any additional cleanup after they click the widget QUIT button.
>>
>> Maybe more to what you are looking for, check out the TLB_KILL_REQUEST_EVENTS and KILL_NOTIFY keyword to the Widget_Base function. You can have the callback routines kill the object when the top level base is destroyed.
>
> I'll look more closely at these but it still looks like the object has to destroy itself (and not just the widget), and I haven't had much luck with OBJ_DESTROY, self. Perhaps I need to also write the wrapper as an object.
>>
>> I have another, more complicated solution if neither of these work for you... Is AstroMap object based on function/object graphics? Or is it made from the old widgets?
>
> No, it uses "classic" widgets with Coyote graphics. --Wayne

I guess I still don't understand the problem. The object is created in association with a widget program, right? The widget program can (probably does) have a CLEANUP routine that is called when the widget dies. All you have to do is find a way to put the object reference into the info structure of the widget. Then, when the widget dies, you get the info structure, find the object reference, and destroy that. There is no "self" there. Just the object reference in a widget CLEANUP routine. I can't imagine how this could cause problems.

```
PRO Widget_Cleanup, tlb
  Widget_Control, tlb, Get_UValue=info
  Obj_Destroy, info.objectRef
END
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

---

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by wlandsman on Wed, 19 Mar 2014 05:36:10 GMT
View Forum Message <> Reply to Message

On Tuesday, March 18, 2014 11:03:59 PM UTC-4, David Fanning wrote:

>
> I guess I still don't understand the problem. The object is created in
>
> association with a widget program, right?

Well, one of the methods of the object creates the widget. So I'd say that the object contains the widget, which is why the widget code refers to self when it needs information from the object (e.g. the top level base ID is stored in self.gui ) I don't have any object reference within the widget method which I can destroy except for self. Thanks, --Wayne

---

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by David Fanning on Wed, 19 Mar 2014 12:23:10 GMT
View Forum Message <> Reply to Message

wlandsman writes:

> Well, one of the methods of the object creates the widget. So I'd say that the object contains the widget, which is why the widget code refers to self when it needs information from the object (e.g. the top level base ID is stored in self.gui ) I don't have any object reference within the widget method which I can destroy except for self. Thanks, --Wayne

I'm thinking something like this:

```
oAstroMap = obj_new('AstroMap')   ;Create the object
oAstroMap.widgetdisplay           ;Create the display widget
```

So, I write the WidgetDisplay method like this:

```
  PRO AstroMap::WidgetDisplay
    tlb = Widget_Base(Title='MyWidget',UValue=self, $
      Kill_Notify='Widget_Cleanup')
    .... ; Doesn't matter what widget I store the object in, $
    .... ; just that it have a kill callback associated with it.
  END
```

And, I write Widget_Cleanup like this:

```
  PRO Widget_Cleanup, widget_that_is_dying
    Widget_Control, widget_that_is_dying, Get_UValue=myObject
    Obj_Destroy, myObject
  END
```

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

---

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by David Fanning on Wed, 19 Mar 2014 12:29:50 GMT
View Forum Message <> Reply to Message

David Fanning writes:

>
> wlandsman writes:
>
>> Well, one of the methods of the object creates the widget. So I'd say that the object contains the widget, which is why the widget code refers to self when it needs information from the

object (e.g. the top level base ID is stored in self.gui ) I don't have any object reference within the widget method which I can destroy except for self. Thanks, --Wayne

> I'm thinking something like this:
>
> oAstroMap = obj_new('AstroMap') ;Create the object
> oAstroMap.widgetdisplay ;Create the display widget
>
> So, I write the WidgetDisplay method like this:
>
> PRO AstroMap::WidgetDisplay
> tlb = Widget_Base(Title='MyWidget',UValue=self, $
> Kill_Notify='Widget_Cleanup')
> .... ; Doesn't matter what widget I store the object in, $
> .... ; just that it have a kill callback associated with it.
> END
>
> And, I write Widget_Cleanup like this:
>
> PRO Widget_Cleanup, widget_that_is_dying
> Widget_Control, widget_that_is_dying, Get_UValue=myObject
> Obj_Destroy, myObject
> END

Of course, in the Cleanup method of the object, you have to be careful not to use code that works with widgets, without first testing if the widget is actually there!

```
PRO AstroMap::Cleanup
  IF Widget_Info(self.gui, /Valid_ID) THEN BEGIN
    .... ; Widget cleanup stuff here
  ENDIF
  ....
END
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by wlandsman on Thu, 20 Mar 2014 22:51:39 GMT
View Forum Message <> Reply to Message

David,

Thanks for this. I think that when destroying an object from a non-modal widget that IDL's automatic garbage collection does not come into play, so that one needs to have proper cleanup methods. And let's just say that my cleanup methods were, um, suboptimal, so it did not appear that the cleanup was working at all. My partial excuse is that automatic garbage collection has made me lazy, but I am also not entirely clear on the concepts.

For example, my map display object includes many other objects which must either be created or passed in via keyword. So my INIT method looks like this:

```
pro mapdisplay::init,oCoord=oCoord
if obj_valid(oCoord) then self.oCoord = oCoord else self.oCoord = obj_new('Coord')
```

Now in the cleanup method, I want to destroy self.oCoord if it was created in the ::INIT, but not if it was passed by keyword. How is the Cleanup method supposed to know which is the case? The only solution I can think of is to add a flag to the object, which can be used by ::CLEANUP

```
pro mapdisplay::init,oCoord=oCoord
if obj_valid(oCoord) then begin
     self.oCoord = oCoord
     self.destroy_oCoord = 0
endif else begin
       self.oCoord = obj_new('oCoord')
     self.destroy_oCoord = 1
endelse
```

Is there a better way than this? Thanks, --Wayne

On Wednesday, March 19, 2014 8:23:10 AM UTC-4, David Fanning wrote:
> wlandsman writes:
>
>
>
>> Well, one of the methods of the object creates the widget. So I'd say that the object contains the widget, which is why the widget code refers to self when it needs information from the object (e.g. the top level base ID is stored in self.gui ) I don't have any object reference within the widget method which I can destroy except for self. Thanks, --Wayne
>
>
>
> I'm thinking something like this:
>
>
>

> oAstroMap = obj_new('AstroMap')   ;Create the object
>
> oAstroMap.widgetdisplay         ;Create the display widget
>
>
>
> So, I write the WidgetDisplay method like this:
>
>
>
>   PRO AstroMap::WidgetDisplay
>
>     tlb = Widget_Base(Title='MyWidget',UValue=self, $
>
>       Kill_Notify='Widget_Cleanup')
>
>     .... ; Doesn't matter what widget I store the object in, $
>
>     .... ; just that it have a kill callback associated with it.
>
>   END
>
>
>
> And, I write Widget_Cleanup like this:
>
>
>
>   PRO Widget_Cleanup, widget_that_is_dying
>
>     Widget_Control, widget_that_is_dying, Get_UValue=myObject
>
>     Obj_Destroy, myObject
>
>   END
>
>
>
> Cheers,
>
>
>
> David
>
> --
>
> David Fanning, Ph.D.
>

> Fanning Software Consulting, Inc.
>
> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
>
> Sepore ma de ni thue. ("Perhaps thou speakest truth.")

---

Subject: Re: How to cleanup an object with a non-modal widget method
Posted by David Fanning on Thu, 20 Mar 2014 22:57:26 GMT
View Forum Message <> Reply to Message

Wayne Landsman writes:

> Thanks for this. I think that when destroying an object from a non-modal widget that IDL's automatic garbage collection does not come into play, so that one needs to have proper cleanup methods. And let's just say that my cleanup methods were, um, suboptimal, so it did not appear that the cleanup was working at all. My partial excuse is that automatic garbage collection has made me lazy, but I am also not entirely clear on the concepts.
>
> For example, my map display object includes many other objects which must either be created or passed in via keyword. So my INIT method looks like this:
>
> pro mapdisplay::init,oCoord=oCoord
> if obj_valid(oCoord) then self.oCoord = oCoord else self.oCoord = obj_new('Coord')
>
> Now in the cleanup method, I want to destroy self.oCoord if it was created in the ::INIT, but not if it was passed by keyword. How is the Cleanup method supposed to know which is the case? The only solution I can think of is to add a flag to the object, which can be used by ::CLEANUP
>
> pro mapdisplay::init,oCoord=oCoord
> if obj_valid(oCoord) then begin
> self.oCoord = oCoord
> self.destroy_oCoord = 0
> endif else begin
> self.oCoord = obj_new('oCoord')
> self.destroy_oCoord = 1
> endelse
>
> Is there a better way than this?

Not to my knowledge. This is the way I do it. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.