Subject: Re: Where is the trick in objects
Posted by David Fanning on Thu, 10 Apr 2003 04:22:31 GMT
View Forum Message <> Reply to Message

Thomas Gutzler (tgutzler@ee.uwa.edu.au) writes:

> I think I ran into a little problem while trying to program my own
> object. I can create it, work with it, and destroy it. But I can't get
> rid of uncleared heap variables.
> I tried a ::KILL method which uses 'heap_free, self' or ptr_free and
> obj_delete on several self.variables and self.objects.
> obj_destroy, myobject doesn't work, too.
>
> I'm kind of clueless.

Oh, dear. Someone needs to write a book on this subject. :-(

> I had a look at FSC_PsConfig which hasn't a single obj_delete and just
> one ptr_free and it uses its own base widget, too.

Well, FSC_PSConfig cheats a little bit and hides a lot of
complexity behind a simple-looking interface. It was designed
to trick people into using objects. :-)

Actually, FSC_PSConfig uses mostly compound widget objects,
which was my first foray into combining objects and widgets.
I wrapped these widget objects into regular procedure calls
so the user wouldn't have to use the Obj_New command and scare
themselves half to death. These objects get destroyed more
or less automatically because there is a KILL_NOTIFY set
for one of the widgets. The object that contains the widget
is destroyed when the widget is destroyed. (We extend this
concept from widgets to almost all objects in the Catalyst
library, which is why we very seldom, if ever, have leaking
memory.)

> My object generates a base widget and some dependent widgets (draw, ..),
> views an image and a polyline and modifies this image. I think, closing
> the base widget without a cleanup-function causes the leak, but PsConfig
> doesn't have a cleanup, too :/

I didn't see a CLEANUP method, but you certainly need one.
I would name your KILL method CLEANUP. The CLEANUP method
will be called automatically when the object is destroyed.

FSC_PSCONFIG *does* have a CLEANUP method to destroy the
one pointer that doesn't get destroyed "automatically".
Basically, the CLEANUP method for the object must perform

OBJ_DESTROY on all objects, and PTR_FREE on all pointers
that are defined in the object class definition structure.

Unfortunately, even adding a CLEANUP method won't solve your
leakage problem because you are making lots and lots of *copies*
of your objects and pointers in your GETPROPERTY and SETPROPERTY
methods!! In fact, these are the methods that are doing you
in. :-)

The problem you are trying to solve with your ALL structure
is that you are trying to do something in an event handler
procedure that really wants to be done in an event handler
method. This is the problem I alluded to earlier today:
How do you combine widget functionality with object methodology?

The easiest way to do this is store the object reference (self)
in the UVALUE of the top-level base. Then make all your events
go to an event handler procedure like this:

```
PRO OBJECT_EVENT_DISPATCHER, event
Widget_Control, event.top, Get_UValue=theObject
theObject -> EVENT_HANDLER, event
END
```

Then, write your event handler method *exactly* like you
would your previous event handler procedure. But now you
have immediate access to all the information in self. There
is no need to copy it any more. :-)

> Anyway, if somebody wants to have a look at the object, it's there:
> http://mugiri-in-au.dyndns.tv:8000/Uni/GUI_CC__define.pro
> Don't be scared of bad style, it's not a final :)

I've seen worse. At least it has comments. :-)

> P.S: I'm also looking for a better way to move the large arrays around.
> I think GetProperty, all=all isn't very fast, is it?

It's slow AND the source of all your problems. I'd get rid of it. :-)

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com

Subject: Re: Where is the trick in objects
Posted by Thomas Gutzler on Thu, 10 Apr 2003 08:30:47 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Thomas Gutzler (tgutzler@ee.uwa.edu.au) writes:
>
>
>> I think I ran into a little problem while trying to program my own
>> object. I can create it, work with it, and destroy it. But I can't get
>> rid of uncleared heap variables.
>> I tried a ::KILL method which uses 'heap_free, self' or ptr_free and
>> obj_delete on several self.variables and self.objects.
>> obj_destroy, myobject doesn't work, too.
>>
>> I'm kind of clueless.
>
>
> Oh, dear. Someone needs to write a book on this subject. :-(

Yeah, I'll do that later :)

>> My object generates a base widget and some dependent widgets (draw, ..),
>> views an image and a polyline and modifies this image. I think, closing
>> the base widget without a cleanup-function causes the leak, but PsConfig
>> doesn't have a cleanup, too :/
>
>
> I didn't see a CLEANUP method, but you certainly need one.
> I would name your KILL method CLEANUP. The CLEANUP method
> will be called automatically when the object is destroyed.

That one was pretty helpfull.

> The problem you are trying to solve with your ALL structure
> is that you are trying to do something in an event handler
> procedure that really wants to be done in an event handler
> *method*. This is the problem I alluded to earlier today:
> How do you combine widget functionality with object methodology?
>
> The easiest way to do this is store the object reference (self)
> in the UVALUE of the top-level base. Then make all your events
> go to an event handler procedure like this:
>

> PRO OBJECT_EVENT_DISPATCHER, event
> Widget_Control, event.top, Get_UValue=theObject
> theObject -> EVENT_HANDLER, event
> END

And this was even more helpfull!

```
IDL> a = OBJ_new('GUI_CC', array) & a->gui
       (... work with the gui ...)
IDL> obj_destroy, a
IDL> help, /heap
Heap Variables:
    # Pointer: 0
    # Object : 0
```

Thanks a lot - again - David.

lucky,
Tom

Subject: Re: Where is the trick in objects
Posted by Pavel Romashkin on Thu, 10 Apr 2003 16:33:49 GMT
View Forum Message <> Reply to Message

Thomas Gutzler wrote:
>
> P.S: I'm also looking for a better way to move the large arrays around.
> I think GetProperty, all=all isn't very fast, is it?

Don't move them around at all. To move them, you need at least twice the memory than the array requires, and possibly three times as much if you are not careful.
Use a pointer to put array onto the heap (if pushing RAM limits, fill the already created pointer array with chunks of data). Then, don't use assignment when you dereference it from the object, rather use in situ dereferencing inside your function call (or method).
The advantage of heap is that the array can stay there forever, even if you quit the process - you can give the pointer to it to another process without ever reallocating memory. This way, I kept arrays as big as 1.2 Gb in RAM and switched from one GUI to another - there is no other way to pass a variable this big as a parameter that I know of.

Cheers,
Pavel
