Subject: Re: large info structure?
Posted by David Fanning on Tue, 02 Oct 2007 15:42:15 GMT
View Forum Message <> Reply to Message

markb77@gmail.com writes:

> I'm writing a widget application which will serve as a data analysis
> platform for scientific image data. The idea is that after loading
> the image, there are many different types of analysis that the user
> may wish to run on it, and this application will support them all.
> There will be an 'Analysis' dropdown menu, for instance, with several
> options. I also want to make it easy to add new analysis methods to
> the program.
>
> I've finished writing the front end and now I'm in the process of
> adding some analysis. What I'm noticing is that the info structure
> that gets passed around between event handlers is getting to be very
> large. Mine is up to 130 variables, at the moment.
>
> Does anyone have a strategy to suggest for dealing with this type of
> situation?

Ahhmumm, may I suggest writing your widget program as an object? :-)

It doesn't solve the "lots of info" problem, but it
does mean you don't have to pass anything around anymore.
It is ALL built right into the fabric of the program.

And the "lots of info" problem eventually gets sorted out
into a "several objects" solution, which tends to compartmentalize
the functionality and keep things from breaking when adding new
functionality.

There is a danger to this kind of programming, however. It is
easy (WAY too easy!) to make your objects "clever". Which is
good, don't get me wrong. But clever objects make it almost
impossible to follow the programming flow. (Have you ever tried
to figure out how something works in the iTool system?) So I would
resist the clever solution whenever possible and write really, really
simple solutions. Six months from now, you will be VERY glad you
did! :-)

Cheers,

David

--
David Fanning, Ph.D.

Subject: Re: large info structure?
Posted by Paul Van Delst[1] on Tue, 02 Oct 2007 16:56:13 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> markb77@gmail.com writes:
>
>> I'm writing a widget application which will serve as a data analysis
>> platform for scientific image data. The idea is that after loading
>> the image, there are many different types of analysis that the user
>> may wish to run on it, and this application will support them all.
>> There will be an 'Analysis' dropdown menu, for instance, with several
>> options. I also want to make it easy to add new analysis methods to
>> the program.
>>
>> I've finished writing the front end and now I'm in the process of
>> adding some analysis. What I'm noticing is that the info structure
>> that gets passed around between event handlers is getting to be very
>> large. Mine is up to 130 variables, at the moment.
>>
>> Does anyone have a strategy to suggest for dealing with this type of
>> situation?
>
> Ahhmumm, may I suggest writing your widget program as an object? :-)
>
> It doesn't solve the "lots of info" problem, but it
> does mean you don't have to pass anything around anymore.
> It is ALL built right into the fabric of the program.
>
> And the "lots of info" problem eventually gets sorted out
> into a "several objects" solution, which tends to compartmentalize
> the functionality and keep things from breaking when adding new
> functionality.
>
> There is a danger to this kind of programming, however. It is
> easy (WAY too easy!) to make your objects "clever". Which is
> good, don't get me wrong. But clever objects make it almost
> impossible to follow the programming flow.

Huh. I read something similar about ruby code the other day also. The real cleverly written ruby codes tends to have many methods that don't seem to actually *do* anything - they just call other methods, which call others, etc..

Of course, I'm mostly still in the monolithic block-o-code stage wrt ruby. :o(

cheers,

paulv

Subject: Re: large info structure?
Posted by Michael Galloy on Tue, 02 Oct 2007 17:52:07 GMT
View Forum Message <> Reply to Message

On Oct 2, 9:10 am, mark...@gmail.com wrote:
> Hi,
>
> I'm writing a widget application which will serve as a data analysis
> platform for scientific image data. The idea is that after loading
> the image, there are many different types of analysis that the user
> may wish to run on it, and this application will support them all.
> There will be an 'Analysis' dropdown menu, for instance, with several
> options. I also want to make it easy to add new analysis methods to
> the program.
>
> I've finished writing the front end and now I'm in the process of
> adding some analysis. What I'm noticing is that the info structure
> that gets passed around between event handlers is getting to be very
> large. Mine is up to 130 variables, at the moment.
>
> Does anyone have a strategy to suggest for dealing with this type of
> situation? I'm worried that this info data will spiral out of
> control. I was thinking that I could break down the info data into a
> bunch of smaller structures, and hold pointers to each of those
> structures in one higher level info structure..
>
> thanks,
> Mark

Not sure what you are storing, but you can eliminate storing any
widget identifiers by giving each widget you want to store a UNAME:

```
myWidget = widget_draw(parent, uname='my_uname', ...)
```

and then using:

```
widgetID = widget_info(event.top, find_by_uname='my_uname')
```

to get the ID later when you need it.

Mike

--
www.michaelgalloy.com

Subject: Re: large info structure?
Posted by markb77 on Tue, 02 Oct 2007 20:25:43 GMT
View Forum Message <> Reply to Message

I like the idea of storing widget names in the UNAME of the widget and then using that name to find the ID... I've actually made that change already and am still left with 130 !

Another question though,

I see many examples of widget event handlers that identify the widget generating the event by checking its UVALUE. It seems redundant to store a name in both the UVALUE and the UNAME of each widget. Is there any disadvantage to using a statement like this:

widget = widget_info(event.id, /UNAME)

to get the source of the event?

thanks,
Mark

Subject: Re: large info structure?
Posted by David Fanning on Tue, 02 Oct 2007 20:48:37 GMT
View Forum Message <> Reply to Message

markb77@gmail.com writes:

> I see many examples of widget event handlers that identify the widget
> generating the event by checking its UVALUE. It seems redundant to
> store a name in both the UVALUE and the UNAME of each widget. Is
> there any disadvantage to using a statement like this:
>
> widget = widget_info(event.id, /UNAME)
>
> to get the source of the event?

The only disadvantage I see is that there are considerably more opportunities to screw up string comparisons in CASE statements and the like then there are with numbers. But if you are careful and ALWAYS stick to a particular case for the names, etc. you should be able to cope.

Cheers,

David

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

Subject: Re: large info structure?
Posted by Paul Van Delst[1] on Tue, 02 Oct 2007 21:08:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> markb77@gmail.com writes:
>
>> I see many examples of widget event handlers that identify the widget
>> generating the event by checking its UVALUE. It seems redundant to
>> store a name in both the UVALUE and the UNAME of each widget. Is
>> there any disadvantage to using a statement like this:
>>
>> widget = widget_info(event.id, /UNAME)
>>
>> to get the source of the event?
>
> The only disadvantage I see is that there are considerably more
> opportunities to screw up string comparisons in CASE
> statements and the like then there are with numbers. But
> if you are careful and ALWAYS stick to a particular case
> for the names, etc. you should be able to cope.

Do you mean a particular case for the the uname/uvalue names, or the case comparison expression (and the like)? It's almost purely a matter of style, but I prefer the latter, e.g.

```
string = 'ThIsCaSe'  ; case irrelevant...whatever looks nice :o)

CASE STRUPCASE(string) OF
  'THISCASE':....
  'THATCASE':....
  ELSE:...
ENDCASE
```

I guess one could argue it's self-documenting and localises any confusion over the convention chosen, but, eh <shrug>.

Anyway...

cheers,

paulv

Subject: Re: large info structure?
Posted by markb77 on Tue, 02 Oct 2007 21:16:54 GMT
View Forum Message <> Reply to Message

On Oct 2, 11:42 am, David Fanning <n...@dfanning.com> wrote:

> Ahhmumm, may I suggest writing your widget program as an object? :-)
>

David - thanks for the advice, btw, but I'm just getting my head
around widget programming.. not quite ready for full on OO programming
yet

Mark

Subject: Re: large info structure?
Posted by David Fanning on Tue, 02 Oct 2007 21:35:22 GMT
View Forum Message <> Reply to Message

markb77@gmail.com writes:

> David - thanks for the advice, btw, but I'm just getting my head
> around widget programming.. not quite ready for full on OO programming
> yet

Well, I can tell you are headed in that direction. :-)

Cheers,

David

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

Subject: Re: large info structure?
Posted by Wox on Wed, 03 Oct 2007 08:06:28 GMT
View Forum Message <> Reply to Message

On Tue, 02 Oct 2007 15:10:57 -0000, markb77@gmail.com wrote:

> I've finished writing the front end and now I'm in the process of
> adding some analysis. What I'm noticing is that the info structure
> that gets passed around between event handlers is getting to be very
> large. Mine is up to 130 variables, at the moment.

How does this "info structure gets passed around between event handlers?" Where is this stored? Are you worried by the program copying this large structure on every event?

I'm not sure what your situation is, but suppose you have the info structure stored in the top level widget, you can access it without copying, like this:

```
pro event_handler,ev
widget_control,ev.top,get_uvalue=infostruct,/NO_COPY
...[handle event: make sure there is no return here!]...
widget_control,ev.top,set_uvalue=infostruct,/NO_COPY
end
```

If you don't like the /NO_COPY idea, you can for example have a pointer (created with ptr_infostruct=ptr_new(infostruct)) to your info structure as UVALUE of the top level widget:

```
pro event_handler,ev
widget_control,ev.top,get_uvalue=ptr_infostruct
...[handle event: reffer to fields as (*ptr_infostruct).(i) ]...
end
```

Subject: Re: large info structure?
Posted by JD Smith on Thu, 04 Oct 2007 22:40:33 GMT
View Forum Message <> Reply to Message

```
>  pro event_handler,ev
>  widget_control,ev.top,get_uvalue=infostruct,/NO_COPY
>  ...[handle event: make sure there is no return here!]...
>  widget_control,ev.top,set_uvalue=infostruct,/NO_COPY
>  end
```

Or, more aptly:

```
; make sure there is no return here, or any error or other stoppage
```

The problem with this NO_COPY method is that if any sort of stop, including an error, happens while the event is being handled, your widget program is "broken" and must be restarted, since the info UVALUE has gone missing. If instead you store state information in a permanent place, like on the pointer or object heap, you can easily patch up your broken event code, retall, and continue where you left off. *So* much nicer for development. I vote to ban /NO_COPY from general recommendation. It's a bit more cumbersome to replace infostruct.x with (*infoptr).x, which is of course why object widgets are so attractive (self.x).

JD

Subject: Re: large info structure?
Posted by Mike[2] on Fri, 05 Oct 2007 15:01:29 GMT
View Forum Message <> Reply to Message

On Oct 2, 5:35 pm, David Fanning <n...@dfanning.com> wrote:
> mark...@gmail.com writes:
>> David - thanks for the advice, btw, but I'm just getting my head
>> around widget programming.. not quite ready for full on OO programming
>> yet
>
> Well, I can tell you are headed in that direction. :-)

Yes - it is inevitable that all will be assimilated!

Mark - A suggestion for those who are interested, but are hesitating to go down the object path:

Pick a project that you are in the process of implementing, or even better, something that you wrote a while back and have to update. Make sure that it is relevant - not just a toy code, but something that you will actually use - some library of non-trivial calculations for example. If you are passing around lots of variables and/or large structures and finding adding new variables to be a real pain, that would be perfect.

Do some reading and studying up on objects (not about IDL objects, but object orientation in general - objects are not about implementation details and syntax, but about design). I got a lot out of reading "Object Oriented Software Construction" by Meyer, but I'm sure there are other things available as well. I think it is key to learn the background without getting locked into some particular language's implementation and syntax.

Then learn the IDL details and syntax for objects and rewrite or

extend your code using them (or whatever language you are using - originally I did this with python).

I'll bet you'll get hooked and learn enough to be fully ready for it.

Mike
