Subject: IDL objects (not object graphics) tutorial?
Posted by Richard French on Thu, 24 Nov 2005 02:33:31 GMT
View Forum Message <> Reply to Message

I'd like to learn how to make use of IDL objects. I'm not ready for object graphics yet, because I'd like to understand INITS and SELF and classes and methods before worrying about viewports and plots disappearing because I am not using the correct projection scheme. I've scoured the web in vain looking for a simple tutorial on how and when to use objects in IDL. I've found a few generic tutorials praising the virtues of object-oriented programming, but almost none of the examples give me any sense of why one would go to the trouble. For example, one tutorial describes an object that can return constants such as the speed of light or Planck's constant, but it isn't obvious to me why this is superior to a simple function that returns clight() or PlancksConst().

What I am looking for is something with a simple application or two in which it is both clear why using objects is superior AND which explains what is meant by self and methods and classes. Without some specific examples to look at, I am having a hard time making sense of the nomenclature or of the value of the approach.

This is prompted in part by David's nifty little pixmap object that I've already made use of in a new program - thanks, David.

Someone must be out there just waiting to get rich writing a book on this topic. The second volume can be about object graphics - I'd settle for the first volume for now - a gentle introduction to objects in IDL. Any suggestions? Thanks!

Dick French

Subject: Re: IDL objects (not object graphics) tutorial?
Posted by b_gom on Thu, 24 Nov 2005 09:04:25 GMT
View Forum Message <> Reply to Message

Hi Richard,

Not knowing what your background is, I'll aim low..

Richard G. French wrote:

> I've scoured the web in vain
> looking for a simple tutorial on how and when to use objects in IDL. I've
> found a few generic tutorials praising the virtues of object-oriented
> programming, but almost none of the examples give me any sense of why one

> would go to the trouble.

'How' is covered reasonably well in the IDL manuals. 'When' is another question altogether.

> What I am looking for is something with a simple application or two in which
> it is both clear why using objects is superior AND which explains what is
> meant by self and methods and classes. Without some specific examples to
> look at, I am having a hard time making sense of the nomenclature or of the
> value of the approach.

Let me try to summarize in a simple-mined, 1:00 AM kind of way:

Think of what problem you want to solve. If it is something like 'calculate such-and-such' or 'perform operation X on an array', then you probably want a simple function of procedure. If you can think of your problem as if it were some sort of abstracted machine or 'object' (like a kitchen appliance or a tree or something), then you probably want to make an Object.

Most IDL widget applications fall into the latter category. Although they can almost always be written using non-OOP techniques, there are often times when the little extra effort to write them as objects pays off. It's hard to actually get a feel for this without trying both techniques first.

Here are two excercises that should illustrate some advantages of OOP:

Q) write a program that acts like a stack of 'things', where a thing is any IDL variable or type. The program should allow you to add things to the stack, pop them off the stack, tell you how many things are on the stack, etc. Also you need to be able to have multiple different stacks of things all co-existing at the same time.

A) Look at the 'linkedlist' object on David Fannings webpage for an object-oriented solution. If you come up with a non-object-oriented solution, let me know.

Q) write a program that acts like a dynamic plot of some data. You need to be able to add lines to the plot, remove them, zoom, change colors, annotate, etc, and all interactively at run-time. Also, your program needs to be simple enough so that other people can call it from their program with a few simple lines of code, and have serveral plots open at one time.

A) Try the non-OOP approach for 15 minutes, then look at David's site. There's probably an object there that has most of what you need, and in

15 minutes you can probably add on a few custom methods to extend the
functionality to do the rest. Chances are, after a few years, you'll
still be adding features to your object and it will do just about
everything you can think of. But, here's the key: after all these
changes, the other programs you wrote that call the object years ago
will still work, and you or others will still be able to extend the
object further or adapt it for other uses.

> Someone must be out there just waiting to get rich writing a book on this
> topic.

Back in 1999, a good general introduction to OOP was being written up.
It's now available here:
http://mindview.net/Books/TICPP/ThinkingInCPP2e.html
This is for C++, but the general principles apply to IDL.

Brad

Subject: Re: IDL objects (not object graphics) tutorial?
Posted by dzarro on Wed, 30 Nov 2005 04:20:46 GMT

View Forum Message <> Reply to Message

Hi,

I have a tutorial at http://orpheus.nascom.nasa.gov/~zarro/idl/objects
that I developed at NASA/GSFC.

Dominic

Richard G. French wrote:
> I'd like to learn how to make use of IDL objects. I'm not ready for object
> graphics yet, because I'd like to understand INITS and SELF and classes and
> methods before worrying about viewports and plots disappearing because I am
> not using the correct projection scheme. I've scoured the web in vain
> looking for a simple tutorial on how and when to use objects in IDL. I've
> found a few generic tutorials praising the virtues of object-oriented
> programming, but almost none of the examples give me any sense of why one
> would go to the trouble. For example, one tutorial describes an object that
> can return constants such as the speed of light or Planck's constant, but it
> isn't obvious to me why this is superior to a simple function that returns
> clight() or PlancksConst().
>
> What I am looking for is something with a simple application or two in which

> it is both clear why using objects is superior AND which explains what is
> meant by self and methods and classes. Without some specific examples to
> look at, I am having a hard time making sense of the nomenclature or of the
> value of the approach.
>
> This is prompted in part by David's nifty little pixmap object that I've
> already made use of in a new program - thanks, David.
>
> Someone must be out there just waiting to get rich writing a book on this
> topic. The second volume can be about object graphics - I'd settle for the
> first volume for now - a gentle introduction to objects in IDL. Any
> suggestions? Thanks!
>
> Dick French

Subject: Re: IDL objects (not object graphics) tutorial?
Posted by Richard French on Thu, 01 Dec 2005 02:36:59 GMT

View Forum Message <> Reply to Message

On 11/29/05 11:20 PM, in article
1133324446.805496.278980@g44g2000cwa.googlegroups.com, "dzarro@yahoo.com"
<dzarro@yahoo.com> wrote:

> Hi,
>
> I have a tutorial at http://orpheus.nascom.nasa.gov/~zarro/idl/objects
> that I developed at NASA/GSFC.
>
> Dominic
>
>

Thanks very much, Dominic! I think I'll be able to make use of pieces of
your OO routines in a new image analysis program I'm working on.
Dick

Subject: Re: IDL objects (not object graphics) tutorial?
Posted by Paul Van Delst[1] on Fri, 02 Dec 2005 15:46:49 GMT

View Forum Message <> Reply to Message

dzarro@yahoo.com wrote:
> Hi,
>
> I have a tutorial at http://orpheus.nascom.nasa.gov/~zarro/idl/objects

> that I developed at NASA/GSFC.

Your webpage is great! The one new thing I learned:

```
   self.ptr=ptr_new(/allocate)
```

can be used to subsequently point to anything without further allocation!? E.g. from your tutorial:

```
   a->set,image
or
   a->set,!d
```

where in the set method, the value is simply assigned:

```
   *(self.ptr)=value
```

Excuse my brain-deadedness, but how is this possible? I looked at the IDL docs but there is (surprise, surprise) no elaboration about this little nugget of information regarding PTR_NEW. Wouldn't susbsequent calls like the above cause a memory leak, e.g.

```
IDL> image=findgen(512,512)
IDL> a=obj_new('data')        ;-- create object variable a
IDL> a->set,image             ;-- insert image
IDL> a->set,!d
```

What would happen to the "image" data?

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC

Subject: Re: IDL objects (not object graphics) tutorial?
Posted by David Fanning on Fri, 02 Dec 2005 16:11:55 GMT

View Forum Message <> Reply to Message

Paul Van Delst writes:

> Your webpage is great! The one new thing I learned:
>
>   self.ptr=ptr_new(/allocate)
>
> can be used to subsequently point to anything without further allocation!? E.g. from your
> tutorial:
>
>   a->set,image

> or
>   a->set,!d
>
> where in the set method, the value is simply assigned:
>   *(self.ptr)=value
>
> Excuse my brain-deadedness, but how is this possible? I looked at the IDL docs but there
> is (surprise, surprise) no elaboration about this little nugget of information regarding
> PTR_NEW. Wouldn't susbsequent calls like the above cause a memory leak, e.g.
>
> IDL> image=findgen(512,512)
> IDL> a=obj_new('data')   ;-- create object variable a
> IDL> a->set,image   ;-- insert image
> IDL> a->set,!d
>
> What would happen to the "image" data?

The deep answers are all contained in the Pointer Tutorial:

   http://www.dfanning.com/misc_tips/pointers.html

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: IDL objects (not object graphics) tutorial?
Posted by Paul Van Delst[1] on Fri, 02 Dec 2005 16:30:07 GMT

View Forum Message <> Reply to Message

David Fanning wrote:
> Paul Van Delst writes:
>
>
>> Your webpage is great! The one new thing I learned:
>>
>>   self.ptr=ptr_new(/allocate)
>>
>> can be used to subsequently point to anything without further allocation!? E.g. from your
>> tutorial:
>>
>>   a->set,image
>> or
>>   a->set,!d

>>
>> where in the set method, the value is simply assigned:
>>   *(self.ptr)=value
>>
>> Excuse my brain-deadedness, but how is this possible? I looked at the IDL docs but there
>> is (surprise, surprise) no elaboration about this little nugget of information regarding
>> PTR_NEW. Wouldn't susbsequent calls like the above cause a memory leak, e.g.
>>
```
>> IDL> image=findgen(512,512)
>> IDL> a=obj_new('data')       ;-- create object variable a
>> IDL> a->set,image            ;-- insert image
>> IDL> a->set,!d
```
>>
>> What would happen to the "image" data?
>
>
> The deep answers are all contained in the Pointer Tutorial:
>
>   http://www.dfanning.com/misc_tips/pointers.html

Ahhh! It's just like a regular IDL variable. D'oh! - now why didn't assume that to be the
case in the first place?!

Very cool. I was treating this sort of thing like I do in Fortran95 where the pointers are
typed (i.e. they can only point to certain data types, kinds, and ranks) and that leads to
a lot of code bloat.

This changes everything. <monty>Excellent</monty> :o)

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC