Subject: Re: status of HDF5 *writing* support in IDL
Posted by Michael Wallace on Tue, 27 Apr 2004 00:08:28 GMT
View Forum Message <> Reply to Message

> Along with many I am sure, I have been waiting patiently for any word on
> support for *writing* HDF5 files in IDL. A major project has been delayed
> in hopes that full HDF5 support would be forthcoming. This issue has come
> up more than a few times in the past week or so as we realize that we can
> delay no longer and now I am getting pissed.

You say this like you think that RSI actually adds useful features to IDL. I'm still waiting for command line arguments and nice-looking fonts in direct graphics. And those aren't the only things missing. At least I can "fix" the command line argument with a nice little Python wrapper.

As far as HDF5 writing goes, RSI did not mention it at all in their last newsletter (http://www.rsinc.com/newsletter/#idlbeta) which listed some of the new features in IDL 6.1. The only thing I know to do is get the HDF5 C library and write a DLM for the functions you need. Why RSI can't do this themselves is beyond me.

-Mike

Subject: Re: status of HDF5 *writing* support in IDL
Posted by Rick Towler on Tue, 27 Apr 2004 20:26:50 GMT
View Forum Message <> Reply to Message

"Michael Wallace" wrote...

>> Along with many I am sure, I have been waiting patiently for any word on
>> support for *writing* HDF5 files in IDL. A major project has been delayed
>> in hopes that full HDF5 support would be forthcoming. This issue has come
>> up more than a few times in the past week or so as we realize that we can
>> delay no longer and now I am getting pissed.
>
> You say this like you think that RSI actually adds useful features to
> IDL. I'm still waiting for command line arguments and nice-looking
> fonts in direct graphics. And those aren't the only things missing.

I don't usually stir the turd and I have to admit that I while I may disagree on certain points, I respect the direction that RSI has taken IDL. My mistake was to assume that since support for reading HDF5 files was added in 5.6, that writing support would be soon to follow.

> As far as HDF5 writing goes, RSI did not mention it at all in their last
> newsletter (http://www.rsinc.com/newsletter/#idlbeta) which listed some
> of the new features in IDL 6.1. The only thing I know to do is get the
> HDF5 C library and write a DLM for the functions you need. Why RSI
> can't do this themselves is beyond me.

I have looked into this, and I also looked into the Java versions in hopes
of using the IDL->Java bridge but the former would be a project unto itself
and the latter approach is plagued by a lack of documentation/examples on
the IDL side.

Have you considered implementing your own home grown anti-aliasing routines
to smooth out those direct graphics fonts (if it is in fact the jaggies that
you despise). I posted some example code a while back...

Thanks for the info.

-Rick

Subject: Re: status of HDF5 *writing* support in IDL
Posted by mmiller3 on Tue, 27 Apr 2004 21:35:00 GMT
View Forum Message <> Reply to Message

>>>> > "Michael" == Michael Wallace <mwallace_spam@spam.swri.edu.invalid> writes:

> I'm still waiting for command line arguments and
> nice-looking fonts in direct graphics. And those aren't
> the only things missing. At least I can "fix" the command
> line argument with a nice little Python wrapper.

Would you post an example of your python wrapper?

Mike

Subject: Re: status of HDF5 *writing* support in IDL
Posted by Michael Wallace on Tue, 27 Apr 2004 22:02:46 GMT
View Forum Message <> Reply to Message

> I don't usually stir the turd and I have to admit that I while I may
> disagree on certain points, I respect the direction that RSI has taken IDL.
> My mistake was to assume that since support for reading HDF5 files was added

> in 5.6, that writing support would be soon to follow.

Don't get me wrong, IDL does do a lot of things right and is good for analysis and visualization. My frustrations with it are mainly because I come from the programming ranks rather than the scientific and I've gotten very use to features that most other languages provide. Also, I've been tasked with the job of shoehorning IDL into places where it doesn't fit so well. For people who want to do things interactively, it's great... maybe that's why it's _Interactive_ Data Language. However, 99% of the things I need to do with it are in the non-interactive realm. And most of these non-interactive features are missing. Anyways....

>> As far as HDF5 writing goes, RSI did not mention it at all in their last
>> newsletter (http://www.rsinc.com/newsletter/#idlbeta) which listed some
>> of the new features in IDL 6.1. The only thing I know to do is get the
>> HDF5 C library and write a DLM for the functions you need. Why RSI
>> can't do this themselves is beyond me.
>
>
> I have looked into this, and I also looked into the Java versions in hopes
> of using the IDL->Java bridge but the former would be a project unto itself
> and the latter approach is plagued by a lack of documentation/examples on
> the IDL side.

Another thing to consider is that the Java version is simply a wrapper around the C version. So, you will still have C code with which to deal. The IDL-Java bridge is a lot easier to set up than a DLM, however and I have the bridge set up on my box and have been able to use it quite successfully. I even managed to get the thing to work on a Macintosh! I was pretty proud of myself since I only touch a Macintosh about once every few months or so. Anyway, the bridge is pretty easy to configure (only one conf file) and it only takes one IDL command to start up Java and one more IDL command to create a Java object.

> Have you considered implementing your own home grown anti-aliasing routines
> to smooth out those direct graphics fonts (if it is in fact the jaggies that
> you despise). I posted some example code a while back...

Right now I don't have the time to implement my own anti-aliasing. Really, it's just an aesthetic issue, so it's not that important -- just a little IDL pet peeve. In fact, I'd be happy just using hardware fonts. However, the problem with them is that the letters can only be printed in their normal orientation. I haven't bothered writing anything which would print letters at some non-zero angle to the horizontal.

-Mike

Subject: Re: status of HDF5 *writing* support in IDL
Posted by Michael Wallace on Tue, 27 Apr 2004 23:24:51 GMT
View Forum Message <> Reply to Message

>> I'm still waiting for command line arguments and
>> nice-looking fonts in direct graphics. And those aren't
>> the only things missing. At least I can "fix" the command
>> line argument with a nice little Python wrapper.
>
> Would you post an example of your python wrapper?

No problem. It's not a high quality Python program, but it gets the job done. Enjoy!

```
#!/usr/bin/python
#
# NDL (Non-interactive Data Language)
#
# Sadly, IDL does not accept command line arguments. This has been the
# bane of my existence for the last couple days. However, there is
# hope and hope comes in the form of this Python script.
#
# This program accepts the name of an IDL procedure, executes the
# procedure and then exits. Additional command line arguments are fed
# as an array of strings to the IDL procedure via the ARGS keyword.
# Therefore, any IDL procedure which needs to accept command line
# arguments needs to define ARGS as a keyword. If there's a procedure
# named IDL_Proc which has the ARGS keyword defined, the following
# command will cause IDL_Proc to run and ARGS will be filled with the
# strings 'arg1' and 'arg2'.
#
# $ ndl IDL_Proc arg1 arg2
#
# When there are no extra command line arguments, the procedure is
# called without the ARGS keyword. This allows you to call code which
# you didn't write or can't edit for some reason. In fact, you can
# actually send an entire IDL command as the first parameter and watch
# it execute. For example,
#
# $ ndl "Print, 'Hello, World'"
#
# will print "Hello, World" to the terminal along with all the standard
# IDL licensing junk you see when starting IDL. If you don't want to
# see all of this, just pipe stderr to /dev/null or some log file. Just
# make sure to not do this if you're debugging something! ;-)
#
# $ ndl "Print, 'Hello, World'" 2> /dev/null
#
```

```
# In short, all this program does is open a pipe to IDL and send an IDL
# statement based on the arguments provided.  That's it.  It's just a
# way to make IDL look like it is accepting command line arguments.
# Nothing more.  Nothing less.
#
#
# March 2004
#


import os
import sys


# Usage statement
usage = "usage: %s idlprog [arg1 [arg2 [ ... ]]]" \
        %os.path.basename(sys.argv[0])


# Check that the name of the IDL program was provided
if len(sys.argv) < 2:
    print usage
else:
    # Open a pipe to an IDL process to write to
    fd = os.popen('idl', 'w')


    # If extra arguments are given, pass them via the ARGS keyword
    if len(sys.argv) < 3:
        fd.write(sys.argv[1])
    else:
        fd.write(sys.argv[1] + ', ARGS = ' + `sys.argv[2:]`)


    fd.close()
```