Subject: Re: recording macros
Posted by Liam E. Gumley on Thu, 19 Jun 2003 19:07:37 GMT
View Forum Message <> Reply to Message

"Ben Tupper" <bentupper@bigelow.org> wrote in message
news:3EF20160.6030007@bigelow.org...
> Hi,
>
> We have an image processing/analysis package - it implements many of the
> methods described in the handy "Image Processing in IDL" book - plus
> other methods/tricks. The package can be run with or without a GUI and
> it is all object based. We have to process a stack of images (up to
> 7200 of 'em), but we only have to set up the image processing steps once
> for each stack.
>
> How might I record the steps the user takes in a macro-like way? I want
> to be able to save the macro to a file for use in later sessions. Has
> this approach been tried in IDL? Any tips or heads-up appreciated.

If it's run from the command line, could you use journalling, i.e.,

IDL> journal, 'my_commands.pro' ; start journalling
IDL> (user enters commands)
IDL> journal ; stop journalling

To replay the session:

IDL> @my_commands.pro

Cheers,
Liam.
Practical IDL Programming
http://www.gumley.com/

Subject: Re: recording macros
Posted by R.Bauer on Thu, 19 Jun 2003 20:39:47 GMT
View Forum Message <> Reply to Message

Dear Ben and Liam,

with journal only the call of the routines could be saved but what is if it
is a widget.

For widgets I have in principle a different solution.

I like to explain how it works in the plot_n widget but this is not an
object.

plot_n writes during execute all changes which are done by the widget into
an idl procedure. This is then executed by the widget by call_procedure and
did the resulting plot.
This could be read in again for initialisation of the widget and so on.

http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/plot_n_dbase.pro.html

Therefore I am using the arg_present trick. plot_n calls it with output
by reading the stored information. Then only the structure is defined and if
a user would call it by plscript then it runs and makes the plot. See
example plscript.pro

I believe you have a structure which you can save first and later compare
for example by diff_struct

http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/diff_struct_dbase.pro.html

```
IDL> d={file:'',set:1}
IDL> master=d
IDL> d.file='test.nc'
IDL> x=diff_struct(d,master)
IDL> help,x,/str
** Structure <840fc2c>, 1 tags, length=12, data length=12, refs=1:
   FILE            STRING    'test.nc'
```

Now you know which things are changed and you have only to setup the makro.
I would suggest to write a procedure as I did. Because then this could be
used without any additional routine to reproduce the result it could be
archived and the users could add idl commands if they like to do it.

```
PRO plscript, output
file=['ghost.nc']
name=['time','F12']
icgs=icgs_read(file,name)
plotprepare, plot
IF ARG_PRESENT(output) EQ 0 THEN plotinit,plot
plot.color=7
plot.psym=4
plot.x_use_units='UTC'
plot.y_use_units='ppt'
plot.xrange=[-9999.0000,6.0000000]
plot.xtitle='time [UTC]'
plot.yrange=[2.0000000,6.0000000]
plot.ytitle='mixing ratio [ppt]'
```

```
IF ARG_PRESENT(output) EQ 0 THEN update_valid_index,icgs,/ignore_quality
IF ARG_PRESENT(output) EQ 0 THEN $
plotxy,plot,x='time',y='F12',icg_struct=icgs
IF ARG_PRESENT(output) EQ 0 THEN plotend,plot
output={file:file,name:name,icgs:icgs,plot:plot}
END
```

regards

Reimar

Liam Gumley wrote:

> "Ben Tupper" <bentupper@bigelow.org> wrote in message
> news:3EF20160.6030007@bigelow.org...
>> Hi,
>>
>> We have an image processing/analysis package - it implements many of the
>> methods described in the handy "Image Processing in IDL" book - plus
>> other methods/tricks. The package can be run with or without a GUI and
>> it is all object based. We have to process a stack of images (up to
>> 7200 of 'em), but we only have to set up the image processing steps once
>> for each stack.
>>
>> How might I record the steps the user takes in a macro-like way? I want
>> to be able to save the macro to a file for use in later sessions. Has
>> this approach been tried in IDL? Any tips or heads-up appreciated.
>
> If it's run from the command line, could you use journalling, i.e.,
>
> IDL> journal, 'my_commands.pro' ; start journalling
> IDL> (user enters commands)
> IDL> journal ; stop journalling
>
> To replay the session:
>
> IDL> @my_commands.pro
>
> Cheers,
> Liam.
> Practical IDL Programming

> http://www.gumley.com/

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: recording macros
Posted by Ben Tupper on Thu, 19 Jun 2003 21:44:22 GMT
View Forum Message <> Reply to Message

Reimar Bauer wrote:
> Dear Ben and Liam,
>
> with journal only the call of the routines could be saved but what is if it
> is a widget.
>
> For widgets I have in principle a different solution.
>
> Now you know which things are changed and you have only to setup the makro.
> I would suggest to write a procedure as I did. Because then this could be
> used without any additional routine to reproduce the result it could be
> archived and the users could add idl commands if they like to do it.
>

>> If it's run from the command line, could you use journalling, i.e.,
>>
>> IDL> journal, 'my_commands.pro' ; start journalling
>> IDL> (user enters commands)
>> IDL> journal ; stop journalling
>>
>> To replay the session:
>>
>> IDL> @my_commands.pro
>>

Thanks Liam and Riemar,

Right now, I don't expect the user to run from the command line since
the GUI is available (and, for the user, less intimidating.)

Since the data format doesn't change (it's always an image) I have an
advantage over more generic data handling. I think that I could add a
RecordMacro method into the event handling to provide. I would have

to have a RECORD property/flag that tells the handler when to record the step and when not to. The behavior I hope to see is similar to that of ImageJ (http://rsb.info.nih.gov/ij/) . When recording, a simple text editor appears listing the steps taken. Here's an example from ImageJ...

> run("Threshold...");
> setThreshold(48, 170);
> run("Threshold", "thresholded remaining black");
> run("Despeckle");
> run("Add...", "value=25");
> run("Measure");

In the IDL case, each set corresponds to a method which could be called in order by CALL_METHOD. The trick, I think, might be the arguments, such as the ADD, value = 25. Perhaps the macro should include the object class, the method and any arguments... I don't know what to do with keywords.

ClassName Method P1 P2 P3 P4 PN
HBB_HISTOGRAM SETTHRESHOLD 48 170
HBB_IMAGE ADD 25

I'll have to think about that.

Thanks again,

Ben

Subject: Re: recording macros
Posted by JD Smith on Thu, 19 Jun 2003 23:38:52 GMT
View Forum Message <> Reply to Message

On Thu, 19 Jun 2003 14:44:22 -0700, Ben Tupper wrote:

> Reimar Bauer wrote:
>> Dear Ben and Liam,
>>
>> with journal only the call of the routines could be saved but what is
>> if it is a widget.
>>
>> For widgets I have in principle a different solution.
>>
>> Now you know which things are changed and you have only to setup the

>> makro. I would suggest to write a procedure as I did. Because then this
>> could be used without any additional routine to reproduce the result it
>> could be archived and the users could add idl commands if they like to
>> do it.
>>
>>
>>> If it's run from the command line, could you use journalling, i.e.,
>>>
>>> IDL> journal, 'my_commands.pro' ; start journalling IDL> (user
>>> enters commands)
>>> IDL> journal ; stop journalling
>>>
>>> To replay the session:
>>>
>>> IDL> @my_commands.pro
>>>
>>>
> Thanks Liam and Riemar,
>
> Right now, I don't expect the user to run from the command line since
> the GUI is available (and, for the user, less intimidating.)
>
> Since the data format doesn't change (it's always an image) I have an
> advantage over more generic data handling. I think that I could add a
> RecordMacro method into the event handling to provide. I would have
> to have a RECORD property/flag that tells the handler when to record
> the step and when not to. The behavior I hope to see is similar to that
> of ImageJ (http://rsb.info.nih.gov/ij/) . When recording, a simple text
> editor appears listing the steps taken. Here's an example from
> ImageJ...
>
>
>> run("Threshold...");
>> setThreshold(48, 170);
>> run("Threshold", "thresholded remaining black"); run("Despeckle");
>> run("Add...", "value=25");
>> run("Measure");
>
>
> In the IDL case, each set corresponds to a method which could be called
> in order by CALL_METHOD. The trick, I think, might be the arguments,
> such as the ADD, value = 25. Perhaps the macro should include the
> object class, the method and any arguments... I don't know what to do
> with keywords.
>
> ClassName Method P1 P2 P3 P4 PN
> HBB_HISTOGRAM SETTHRESHOLD 48 170
> HBB_IMAGE ADD 25

>
> I'll have to think about that.
>

If these are objects, why not encode the macro history within the object itself? I can imagine keeping, e.g., self.history, as a pointer to a list of structures like:

```
st={MACRO_ENTRY, $
   method: '', $
   args: ptr_new(), $
   kwds: ptr_new(), $
   undo_data: ptr_new()}
```

with args as a pointer to a list of pointers to individual arguments, and kwds as a pointer to an _EXTRA style structure, ala {KEYWORD: value, KEYWORD2: value2}. Re-running a macro is then as easy as (assuming procedure-methods only):

```
n=ptr_valid(macro.args)?n_elements(*macro.args):0
 extra=ptr_valid(macro.kwds)?*macro.kwds:{____NOTAKEYWORD:0b}
case n of
  0: call_method,macro.method,self,_EXTRA=extra
  1:  call_method,macro.method,self,*(*macro.args)[0],_EXTRA=extra
  2:  call_method,macro.method,self,*(*macro.args)[0],*(*macro.arg s)[1], $
            _EXTRA=extra
  ...
endcase
```

Of course, you have to use an ugly CASE to treat differing numbers of parameters, and you'll probably run out of steam before reaching the 64000 parameter limit, but in practice, 10 ought to do it. The other option which avoids this hackery is to use EXECUTE with a hand-built statement, but this is slow(er), and precludes the use of, e.g., large arrays of input (without lots of temporary variables). A hybrid would be to build an execute statement using the argument data from the pointer heap (though there's also a finite limit on the length of the EXECUTE string):

```
void=execute('self->'+macros.method+','+strjoin('*(*macro.args)['+ $
        strtrim(indgen(n),2)+']',',')+',_EXTRA=extra')
```

If your methods are meaty and take a while to run, the EXECUTE overhead won't be noticeable. Another clever idea to go along with this would be to bundle a snapshot of the data being processed before the macro was completed with the macro entry itself (UNDO_DATA above), for easy UNDO/REDO functionality. For popping macro entries off the history and removing them, HEAP_FREE is your friend.

You can save either just the self.history structure to an IDL
savefile, or the entire object itself. I'd vote for the object, since
then you get all the data in its present state. If you want to save
sans the UNDO data to save space, just "prune" it out before saving by
setting to a null pointer, and restore afterwards.

JD
