Subject: Re: Saved Visualizations
Posted by SonicKenking on Sat, 05 Feb 2011 07:22:10 GMT
View Forum Message <> Reply to Message

> Well, since you are interested in noodling around with
> save files, why don't you write a routine that can
> allow the user to view and unpack the commands in the
> save file?
>
> There is already a preliminary "viewer" in the commands
> LIST method. But I can see a program that "unpacks" a
> command to produce variables at the main IDL level containing
> all the data. Perhaps each command could be put into a
> structure variable that was returned to the main IDL level.
>
> cmd -> {p1:cmd.p1, p1:cmd.p2, ..., NLevels:cmd.extra.nlevels}
>
> Such a thing could be very useful because it would
> allow users to "recover" data from previous visualizations.
>

Hi David,

Following your suggestions, I came up with a new method for the FSC_Window_Command class for creating the command struct variable in IDL main level. It is written using the LIST method as the template.

I also added an keyword option to FSC_CmdWindow::ListCommand. The method will also create the struct variable when the option is turned on.

The command struct variable has the data structure as follows:
cmdStruct = {command: 'cgContour', nparams: 1, type: 0, p1: data, keywords: {nlevel: 10, fill: 1}}

I also wrote a small procedure, which takes the cmdStruct as input parameter and execute it. The struct variable can be saved as IDL source files (*.pro files) with proper output format. So it almost sounds like another way (ASCII files) for saving the visualization. It is quite interesting.

Here is the method, FSC_Window_Command::CreateCommandStruct, for create the command struct variable in IDL main level.

```
PRO FSC_Window_Command::CreateCommandStruct, structName, Quiet=quiet

   Compile_Opt idl2
```

```
    ; Error handling.
    Catch, theError
    IF theError NE 0 THEN BEGIN
        Catch, /CANCEL
        void = Error_Message()
        RETURN
    ENDIF

    ; Struct variable name
    IF N_Elements(structName) EQ 0 THEN structName='cmd'

    cmdString = self.command
    cmdStruct = Create_Struct('Command', cmdString, 'nparams',
self.nparams, 'type', self.type)
    CASE self.nparams OF
        0:
        1: cmdStruct = Create_Struct(cmdStruct, 'p1', *self.p1)
        2: cmdStruct = Create_Struct(cmdStruct, 'p1', *self.p1, 'p2',
*self.p2)
        3: cmdStruct = Create_Struct(cmdStruct, 'p1', *self.p1, 'p2',
*self.p2, 'p3', *self.p3)
    ENDCASE
    IF Ptr_Valid(self.keywords) THEN BEGIN
        cmdStruct = Create_Struct(cmdStruct, 'keywords',
*self.keywords)
    ENDIF

    ; Copy the variable to the MAIN level
    (Scope_VarFetch(structName, /Enter, Level=1)) =
Temporary(cmdStruct)
    IF NOT Keyword_Set(quiet) THEN $
        PRINT, 'Created command struct variable ', structName, ' in
IDL $MAIN level.'

END

 ############################################################## ################
```

Here is the modified version of FSC_CmdWindow::ListCommand for adding the option to create the command struct variable. The changes are only adding three new lines.

```
PRO FSC_CmdWindow::ListCommand, cmdIndex,
CREATECOMMANDSTRUCT=createCommandStruct

    ; List the commands in the command window.

    Compile_Opt idl2
```

```
   ; Error handling.
   Catch, theError
   IF theError NE 0 THEN BEGIN
      Catch, /CANCEL
      void = Error_Message()
      RETURN
   ENDIF

   createCommandStruct = Keyword_Set(createCommandStruct)

   ; How many commands are there?
   count = self.cmds -> Get_Count()

   IF N_Elements(cmdIndex) EQ 0 THEN BEGIN
      FOR j = 0, count-1 DO BEGIN
         thisCmdObj = self.cmds -> Get_Item(j, /DEREFERENCE)

         ; Preface the commands with their index number.
         thisCmdObj -> List, StrTrim(j,2) + '.'

         ; Create the command struct
         IF createCommandStruct THEN thisCmdObj ->
CreateCommandStruct, 'cmd' + StrTrim(j,2)
      ENDFOR
   ENDIF ELSE BEGIN
      IF cmdIndex LT (count-1) THEN BEGIN
         thisCmdObj = self.cmds -> Get_Item(cmdIndex, /DEREFERENCE)

         ; Preface the commands with their index number.
         thisCmdObj -> List, StrTrim(cmdIndex,2) + '.'

         ; Create the command struct
         IF createCommandStruct THEN thisCmdObj ->
CreateCommandStruct, 'cmd' + StrTrim(cmdIndex,2)
      ENDIF ELSE Message, 'The command index is out of range of the
number of commands.'
   ENDELSE

END
```

################

The last one is the small procedure for execute the command in the struct variable. Note that the winid keyword does not work. The plot always goes into a new cgWindow. I am not sure why it is the case.

```
PRO cgRunCmdStruct, cmd, WinID=winid

   Compile_Opt idl2

   ; Error handling.
   Catch, theError
   IF theError NE 0 THEN BEGIN
      Catch, /CANCEL
      void = Error_Message()
      RETURN
   ENDIF

   void = Where(Tag_Names(cmd) EQ 'KEYWORDS', count)
   IF count NE 0 THEN extraKeywords = cmd.keywords

   CASE cmd.nparams OF
      0:
      1: cgWindow, cmd.command, cmd.p1, Method=cmd.type,
WinID=winid, _Extra=extraKeywords

      2: cgWindow, cmd.command, cmd.p1, cmd.p2, Method=cmd.type,
WinID=winid, _Extra=extraKeywords

      3: cgWindow, cmd.command, cmd.p1, cmd.p2, cmd.p3,
Method=cmd.type, WinID=winid, _Extra=extraKeywords
   ENDCASE

END
```