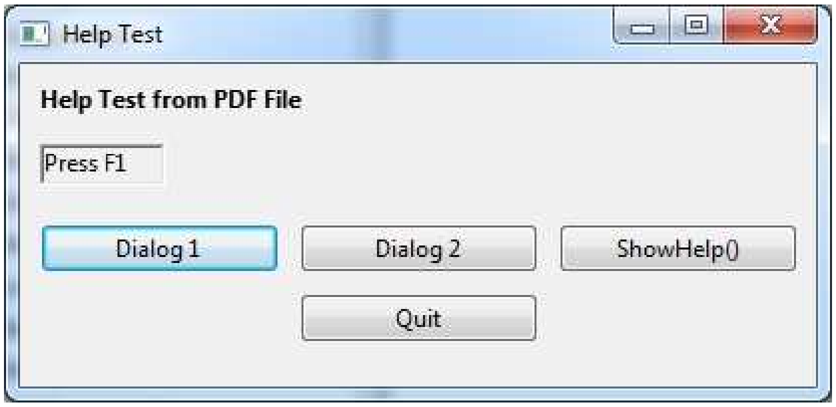
Help Test using PDF file

Table of contents

Help Test



Hi All,

Once upon a time, Microsoft has decide that .CHM files would not be visible anymore from a network drive because it may contain harmful scripts. But those same .CHM files could still be visible from a local drive. Hum... I don't understand why it would be harmful from a network drive and not from a local drive. After all it's the same file. But let's not talk about that.

I have applications that needs an interactive and complete user manual. They are sometime, relatively small and sometime, relatively large. The method of displaying a .CHM was a nice and easy way to accomplish this. Unfortunately it's not available anymore and I didn't find anything that could replace it.

Some may say that I can create a PDF file as a reference guide to my application to solve the situation. Yes, your right. I could do so and I have done so. But I had remarks by users that say it's a tedious task to, open Windows Explorer then find the application folder then open the PDF file then search the PDF for the information they are looking for, either in the topic browser pane or from within the content pane.
I must agree that especially for someone who is not at ease with all this manipulation, it can be discouraging.
So...

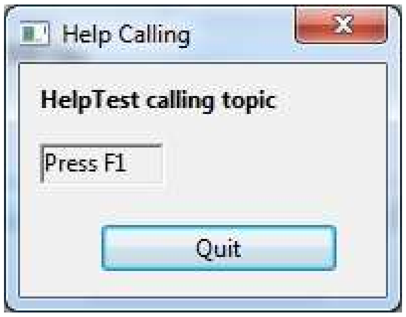
Not everyone is at ease working with computers, we have to think of that, and offer a fast and easy way to access the provided user guide. So...

I thought it would be nice to use a PDF file, but also reproduce WinHelp behavior. There may be many way of doing it, but I have found a way to reproduce this quite well. Since WinHelp and PDF viewer applications have similar ways of displaying their content (I mean a Topic pane on the left of the frame and a content pane on the right of the frame), maybe I could try to do this for the user. So...

The trick is to call a PDF viewer with some parameters that will enable the viewer to seek a specific topic and position the document so that the associated content information is displayed fast and easy.
MiniGUI has the power to do this and I had to try it. So...

Press <Esc> key to return to the main window, then click on **Dialog 1** button.

Help Calling



Hi again,

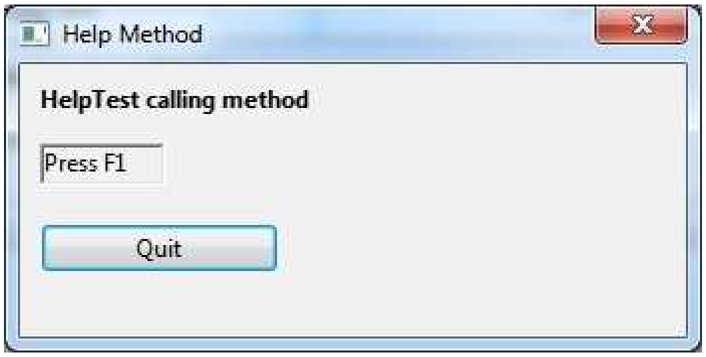
Like I said, I wanted to reproduce the behavior that we all know from WinHelp when displaying help files like .CHM. That being said, I needed to find a PDF viewer that would allow to launch the search of a topic from a command line parameters. Since the PDF viewer was to be an external application launch with ShellExec() it needed to offer some search options and maybe a bit more. So...

I currently use SumatraPDF for previewing reports and lists in many of my applications. This viewer had been of many service in all kind of situation. It is portable, so it can be installed in it's own folder within your application main folder, it is freeware, as long as the GNU license is respected (very important), it is lightweight, it has an integrated search engine, it can open multiple document, but most of all it has a list of command line options that are interesting. So...

I also currently use HelpNDoc as an help file editor, which can generate not only CHM document, but PDF document as well. It also allows to integrate graphics and has many of the nice things that a word processor offers to make nice documents. So...

*Go back to the main window, but leave the PDF viewer open then click on **Dialog 2** button.*

Help Method



Hi,

You're still there! Good, that means you have an interest for what I've done.

Here it is:

I create the help document with HelpNDoc then save it in PDF format to my application folder.
I incorporate SumatraPDF with my application installation so that SumatraPDF has it's own folder within my application folder.
And then whenever I press F1 key from anywhere in my application I get the PDF viewer to open and position itself to the right topic content.
Moreover I get a look similar to WinHelp look with search option and possibility to browse within the PDF document without leaving it.
Isn't that great...
I think so too...

Of course my application has to be modified for this to work correctly, but it's quite simple.
First you need to add the following line at top of your main PRG:

```
   #include "i_hmgcompat.ch"    // For GetFormNameByIndex()
```

Then you need to install an event handler at beginning of main PRG for the F1 key to operate everywhere in your app:

```
   InstallEventHandler("KeyTrap")
```

Next you need to create the trapping function that will reassign the F1 key according to the window or form name:

```
   //-----------------------------------------------------------------------
   //  KEY HANDLER
   //-----------------------------------------------------------------------
   function KeyTrap (hWnd, nMsg, wParam, lParam)
   local cFormName

   if nMsg == 6 // WM_ACTIVATE
      cFormName := GetFormNameByIndex(GetFormIndexByHandle(hWnd))
      _DefineHotKey(cFormName,, VK_F1, {|| ShowHelp()})
   endif

   Return (NIL)
```

Now the function that will call the PDF viewer with the topic to be displayed:

```
   function ShowHelp (cTopic)
   local cParams

   if cTopic == NIL
      cTopic := ThisWindow.Title
   endif
   cTopic  := iif(cTopic == NIL, "", '"'+ cTopic +'"')
   cParams := '-esc-to-exit -named-dest '+ cTopic +' -reuse-instance HELPTEST.PDF'
   ShellExecute(, "OPEN", "SumatraPDF\SumatraPDF.exe", cParams,, 1)

   Return (NIL)
```

The real trick reside in the command line options (switches) that SumatraPDF supports. Though there are more options available, those explained here did the job:

-esc-to-exit --> Allows the user to quit the help file (SumatraPDF viewer) by simply using the <Esc> key.

-named-dest --> This option tells SumatraPDF what to search for in the Topic table (the left pane).

-reuse-instance --> This one tell SumatraPDF to reuse the same opened instance of the program and update the view with the new searched topic.

Go back to the main window, but leave the PDF viewer open then click on ***Restrictions*** *button.*

Restrictions

Well, I'm happy to see that you are still with me.

In the last topic I have shown how simple you can reproduce and control help files using SumatraPDF viewer. But wait, there are restrictions. Well there is one restriction, but there is also more to this.

In order to fetch the right topic content of the PDF file, it needs to be an exact match. SumatraPDF will search the **-name-dest** value in the topic table. If it finds it, that content will be displayed for the user. If not, then it will position to the topmost topic.

In the beginning I had developed a routine that would retreive the name of the window (form) then seek into database indexed on the window names, then use an associated window description to be passed as **-name-dest** value to accomplish the work. Doing so meant that a database containing the window names and topic descriptions would need to be maintained by someone to cross reference both information. Then I thought, Nah! Too complicated and not practical. Instead I used the title of the window (form). This way it didn't need to have a cross reference database which inturn means no maintenance. The window title would be the reference to be passed as **-name-dest** value. And it works...

But wait there's more. You can also use the same function and have the same effect without using the F1 key. Simply call **ShowHelp(cTopic)** from a button action or any other control action clause.

Ex.:
```
@ 80, 270 button oBtnDirect      ;
          caption "Restrictions" ;
          width 120 height 25    ;
          action {|| ShowHelp("Restrictions")}
```

That's how you've been brought you to this topic.

You can also use **ShowHelp("")** then SumatraPDF will bring you to the topmost topic of the PDF help file.

The way it work:

If cTopic is NIL, ShowHelp() will fetch the window title using **cTopic := ThisWindow.Title**
If cTopic is empty or not it will pass the received **cTopic** string

But wait there even more. Throughout this demo I have told you not to close the PDF viewer in order to demonstrate that SumatraPDF was capable of updating the same instance instead of opening a new one for every call made by ShellExec().

That's nice, but what about if I close my application without closing SumatraPDF. Well I also thought about that by auto closing SumatraPDF when quitting the application making SumatraPDF entirely part of the application. Look at the code:

```
function Quit ()
local z
local aProcess

if !MsgYesNo("Do you wish to quit the Application ?", "Help Test from PDF")
   Return (NIL)
endif

aProcess := GetProcessesNT()
for z = 1 to Len(aProcess)
   if (ValType(aProcess[z]) == "C" .and. "SUMATRAPDF.EXE" $ Upper(aProcess[z]))
      KillProcess(aProcess[z -1])
   endif
next z
```

```
   ReleaseAllWindows()

   Return (NIL)
```

For that you will need to add MiniGUI\Lib\ProcInfo.lib in you link. It works nicely.
You don't believe me ! Ok.
Go back to the main window, but leave the PDF viewer open then click on **Quit** button.

Oh ! Before you go. I did not experiment with other PDF viewer and other Help editor because those two software that I've been using for a long time did a great job for what I wanted to do. So...

Feel free to experiment with other software if you like and feel free to experiment with my demo to fit your needs and wishes. Over the years many people help me through MiniGUI forum, so it's my way of saying thanks to the community. If this method of managing your application help files works for you then you see me happy of being a small contribution to help.

Regards
Gilbert Vaillancourt