

Folders

- by James Parr -

Intro:

I hate VBXs. They do nice things, and I have a lot of third party VBXs; but I hate having to distribute additional stuff with my apps, and worry about compatibility and upgrades.

To this end, I have tried to duplicate a lot of fancy functionality in straight code. Here is my first success.

The Folders module provides a rolodex like folder tab system. By simply creating an array of `Frame`s, you can have a completely automated, Microsoft like, folder tab dialog box. If you don't like the way they look, you can always modify the code to fit your needs.

That being said, here is how it works:

Setting Up:

What the Folders module needs to function is an array of `Frame` controls, and a `PictureBox`. Below is how to configure each item:

`Frame(0)`

Name: Whatever

Location: Wherever, sets the location for all folders

Size: What you want all the folders to be sized as

Caption: What you want the first tab to read

BackColor,ForeColor: What you want all the folders to look like

Font: What you want the tabs to look like

`Frame(1-n)`

Name: Same as `Frame(0)` dummy, it's an array.

Caption: What you want the tabs to read.

Location,Size, etc.: Doesn't matter, they will be set to match `Frame(0)`

`PictureBox`

Doesn't matter, all properties are set by the module

In each `Frame`, put the controls that you want to appear on that folder. Remember that accelerator keys defined for a control will not be active if that folder is not at the front.

Starting the Program:

The best place to put the first line of code is in the form's `Load` procedure. The command is simple:

```
Success% = DefineFolders(4, Frame(0), PictureBox)
```

The first parameter (4) indicates the number of tabs across you want to display (sets the variable `VisibleTabs`). If there are more folders than `VisibleTabs` then there will be multiple rows of tabs. If you set this value to zero (0), it will automatically be set to the number of folders.

The second parameter (`Frame(0)`) is the first element in the `Frame` array. The program uses this control to track down all the other elements of the array.

The third parameter (`PictureBox`) is the name of the `PictureBox`. The program uses

this reference to manipulate the Picture to contain the tab pictures. Again, the program sets all the appropriate settings of this control.

`DefineFolders` returns TRUE if it did not encounter any problems.

You will need to add a single command to the `MouseDown` event of the `PictureBox`, in order to process the user's clicks on the tabs.

`FolderClick Button, X, Y`

This line calls the `FolderClick` procedure with the button used and the X and Y coordinates of the mouse. The `FolderClick` procedure calculates which button was pushed, and makes that folder active.

Other optional procedures that you can use are:

`NextFolder & PrevFolder`

These procedures make the next or previous folders active. They circle around the end to the beginning and vice versa.

It is a handy trick to put these functions in the `KeyDown` procedure of the form, and make them respond to a `PGUP` and `PGDN`. For example:

`If KeyCode = &H21 Then PrevFolder`

`If KeyCode = &H22 Then NextFolder`

In order for this to function, you must set the form's `KeyPreview` property to TRUE.

`GotoFolder n`

This allows you to jump to any folder. If the number is less than zero (0) or greater than `NumFolders` then it will be ignored.

`TightenForm`

This procedure moves the folders and associated parts to the top left corner of the window. Then it reduces the size of the window to just fit the folders. If you don't intend to put anything else in your dialog but the folders, this procedure will shrink everything to fit the window exactly. (This was one of the toughest procedures to code because I need to take into account menu and caption heights as well as border widths.)

Modifying the Code:

Well, most of the code works just fine. The only place I can see the need to modify is the drawing of the tabs. If you would prefer a 3D look, or something like that, you can easily change the `DrawTab` procedure.

The `DrawTab` procedure calculates the Left, Right, Top, and Bottom of a tab, then uses the `Line` method to draw the tab itself. Modifying this code allows you to make the tabs look like you want them to.

The rest of the code has been commented fairly well so you can flow through it and see how it works.

Problems and Omissions:

- o If the text is too big for the tab, it overwrites it.
- o If you put the folders too close to the top of the window and you have multiple tab rows, you will lose the top rows off the top of the window
- o There is no way to inactivate a tab (You can set a `Frame`'s `Enabled` property to FALSE in order to inactivate its contents, but you can still select that folder).

Comments:

Feel free to let me know how it works for you.

If you improve it, let me know what you did.

My next project is a better button bar. If you have any coding suggestions, please let me know.

If you want to talk Harleys or Jethro Tull, drop me a line.

Reach me at:

CIS 73312,3615

- James Parr -

5/31/94