FoxPro Grid Hovering Class (and more...)

Introduction

What is hovering? According to the Cambridge Dictionary it means: "*to put the cursor on a computer screen in a particular place without clicking on it*". Thefreedictionary.com states: "*to position a pointer over an object or area of the screen, causing a pop-up box to appear or other change to occur*". You probably do this every day, just by moving your mouse pointer on your desktop, in Windows and in various applications and online in your browser.

Indeed, when you pay attention, you will find hovering in all sorts of situations, applications and operating systems. Just open Windows Explorer or open a webpage and you will see that the content is often shown according to the mouse pointer's position.

We understand why users like this, because this is a very direct and interactive way of using the user interface and making the mouse movement visible. And of course this it works with a touschscreen too, with your finger instead of a mouse.

How about hovering in Visual FoxPro?

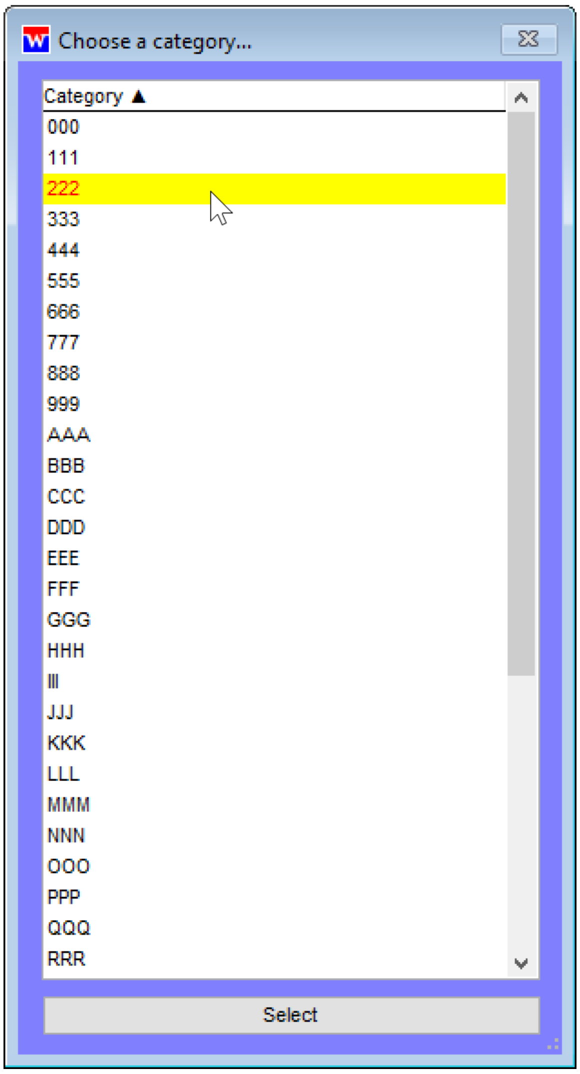
When using Microsoft Visual FoxPro (a language no longer actively supported by Microsoft, but nevertheless still with a very active community and lots of users and developers worldwide) there is not much that will happen when hovering. OK, the FoxPro menu has hovering by default and enabled buttons change when hovered. You can perhaps look at the tooltips as a kind of hovering. And with grids the header will change a bit when hovered. I'm still not very impressed.

If you select a file from your VFP application you will probably see a Windows form being called, with hovering!

Why Microsoft never added hovering to other form elements, like grids, remains a mystery for me. After all the "visual" part of Visual FoxPro could have offered more than just basic Windows functionality.

We could see some use for hovering a grid when showing for example a selection form in a larger application, like shown below.

This form would have a grid, showing the records to choose from. We use many of these forms in out applications. In our case you can select by (1) keyboard ENTER, (2) double clicking the grid and (3) clicking the SELECT button on the form.



Searching the internet with my friend Google (and even trying AI for this) I didn't find very elaborated ways to get hovering working for grids.

However I did find a few suggestions that someone had in mind but had never tried themselves. But this enabled me to start thinking and trying and the well respected users of Foxite and Tek-Tips may have been wondering about some of my very weird questions for bits and pieces I tried to develop this class.

No matter what you try and look for, Visual FoxPro has no option to activate hovering in grids.

Untill now!

Development

Over the last months I have worked on and off towards a FoxPro class that can easily be added to a form (including existing forms). Our forms already had certain features, like (1) grids with highlighting (pick your own colors), (2) sorting grids ascending or descending per column by clicking on the grid column's header (with a changing picture to make the sorting order visible), (3) an incremental search for each grid's textbox type CurrentControl, using a 3rd party class (ingrid.scx) and (4) saving each form's desktop position and size (per user) to our database. And almost all our forms with grids are sizable (between practical limits).

Our grids now use the newly added class "hovercontainer". This adds certain features and has some prerequisites as well. Let's take the prerequisites first, as they more or less determine if this class fits your (and our) needs.

What is needed:

- a form with a grid
- a table or cursor with an active index

The grid’s name can be any valid FoxPro name, as the class looks up the actual grid name. If your table or cursor has no index, you can probably make one in you form’s LOAD event. If there is no active index, hovering will be disabled and an “Oops” message will be shown to make sure you will do your homework!

In our applications we have a “user” table. In this table we can enable and disable the hovering per user. In other words: it’s absolutely not mandatory for your users to use the hovering if they don’t like it, start making errors or become insecure.

Incremental Search

Since our HoverContainer class broke our existing code for Incremental Search (per column) we have added our own incremental search function.

The incremental search is searching in the active column. If you have a grid with multiple columns, click the column’s header to select and click the header once more to sort that column in opposite order. The active column and the sorting order is shown with a small black triangle pointing UP or DOWN in the grid’s header. The grid sorting functionality and the placement of the black triangles is not part of the HoverContainer class, but the sample form will show how this works.

In this version we made some choices that you may or may not like. We simply asked our users what they preferred. So these choices may have a “Dutch” state of mind.

First of all we have decided to simplify “special” characters to “normal” characters. To give an example: all of the following characters aAàÀáÁâÂãÃäÄåÅ were simplified to A.

Secondly we have allowed all other characters, like @#$&.

Finally we have decided (for now) to search only from left to right. So when a search term is in the middle of a field, it will not show up as a result.

After each keyboard input this function waits for 1 second for any new input. When this time has expired, a new search will start with new keyboard entry.

In order to get the incremental search working with keyboard input we have to forward the input to our HoverContainer by using:

```
ThisForm.hovercontainer1.IncrementalSearch(nKeyCode)
```

This command is placed in the grid’s KeyPress event.

Last but not least we changed our incremental search in such a way that it remains working, even if you switch off hovering! For our applications this means we no longer have to use ingrid.vcx to get an interactive search function (when the hovering is disabled).

AutoScrolling

Enabling hovering on a grid is only a part of the solution we present here. We combine this with our "AutoScrolling" function. This function is active when there are more records than can be shown on the grid.

When we hover the grid and reach the very top position on the grid, it will start scrolling upwards and when we hover to the very bottom position it will start scrolling downwards. If the first (or the last) record becomes visible while scrolling, the scrolling will stop.

Of course the mouse wheel is still active and you can still the vertical scrollbar, so you can continue scrolling as you're used to.

AutoScrolling is always active by default.

MultiSpeedScrolling

By using the HoverContainer we learned that it would be nice and practical to use autoscrolling with a slow start. In other words, scrolling starts with a small interval between record changes for a limited number of records.

The number of records can be set in a variable or form property. By default, we show 10 records with a small interval, then 20 records with a smaller interval and the remaining records with no interval at all (full speed ahead!).

Every time the BottomContiner or the TopContainer get activated the MultiSpeedScrolling will be reset.

The MultiSpeedScrolling and the initial number of records with slower scrolling can be set using variables. So it's possible to disable MultiSpeedScrolling or set the number of records with slower scrolling.

Ho Ho Ho!

When you want to disable hovering, for example because the mouse pointer is above the wanted record, you can simply click on the grid with the left mouse button. The hovering will stop. If you want you can still select another record shown on the grid, just by clicking or by using the keyboard's arrow keys. For selection you can use (1) keyboard ENTER, (2) doubleclick and (3) the form's SELECT button.

When you want to re-enable hovering, you can right click on the form's select button. All hovering properties and functionality will be restored and activated.

How does it work?

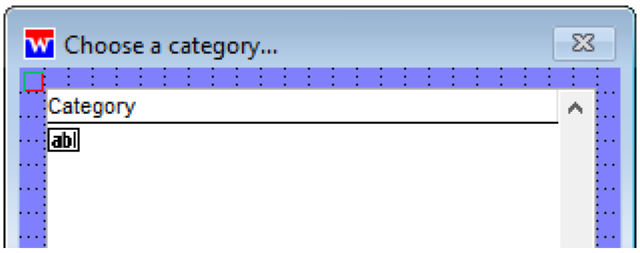
Without going too much into the details this is what happens. The form's grid is fully covered with a container (our HoverContainer). As a user you won't even notice it's there. Movements of the mouse over the container are translated to the grid. This is the basic hovering. This class allows form and grid resizing and respects the grid's headerheight and rowheight properties.

For the AutoScrolling function I added two containers to the HoverContainer. These containers are transparent too and are placed on top of the HoverContainer. A container called TopContainer, to allow scrolling when the mouse enters the top record position of the grid. The BottomContainer takes care of scrolling when the mouse enters the bottom record position of the grid. Once you have tried the HoverContainer you immediately feel and understand how this works.

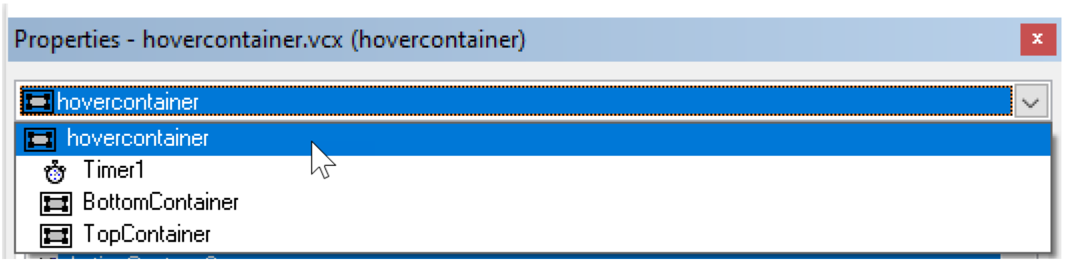
For the Incremental Search function I added a timer to make sure we wait no longer than one second for any new input.

Note: When developing in VFP's IDE all three containers are made visible, so you can see they are there and that the class is activated.

When developing you will only see a very small square symbol, showing that the class is added to a form. It's by default only 10 x 10 pixels in size, so it's usually not a big deal placing this somewhere on a form. Here you can see the class in the upper left corner, just below the form's icon.



This is the class structure as described above:


Properties - hovercontainer.vcx (hovercontainer)

hovercontainer

hovercontainer
Timer1
BottomContainer
TopContainer

Since a container does not have a KeyPress event I added a small procedure to our library. You can put this procedure wherever you think it's adequate. This can be in your own library of prg's or in your start.prg.

```
*****************************************************************
* KeyboardEnter is being called by hovercontainer.mousenter     *
* when using the ENTER key, hovercontainer does not have a      *
* KeyPress event, therefore we guide ON KEY LABEL ENTER to      *
* Grid1.DblClick with Form.NAME as parameter. GridName is read  *
* from the form's hovercontainer.                               *
*                                                               *
* © Gerrit J. Broekhuis                   last edit 27-07-2023  *
*****************************************************************
PROCEDURE KeyboardEnter
PARAMETERS cForm
LOCAL lcGridName
lcGridName = &cForm..hovercontainer1.GridName
&& WAIT WINDOW "Form     = " + TRANSFORM(cForm)
&& WAIT WINDOW "Gridname = " + lcGridName
TRY
	&cForm..&lcGridName..DblClick
CATCH
	WAIT WINDOW "ENTER for form " + cForm + " is not possible!" timeout(1)
ENDTRY
ENDPROC
```

By default this procedure is used from the HoverContainer's MouseEnter event, due to this code.

```
ON KEY LABEL ENTER do KeyboardEnter with _screen.ActiveForm.name
```

It is possible to disable this function, by setting the property HoverContainer.EnableKeyboardEnter to .F..

Note: When calling a form with the HoverContainer class, make sure you use the name clause when calling the form. Keep it smart and simple and use the form's name in the NAME clause, like for example:

```
DO FORM "maindesc.scx" NAME MAINDESC      && NAME clause is required
```

If you omit the NAME clause the procedure KeyboardEnter will be unable to find the object.

Form enhancements

Over the years we have become used to forms with grids that are sizable. I guess most developers will use forms and grids like that.

When working with the HoverContainer class I learned that just resizing forms pixel by pixel gives ugly results and extra complications when using the BottomContainer for scrolling down. The solution is to scroll using grid and form heights that fit entire records. This means the heights are measured, depending on the grid's rowheight setting. So with a rowheight of for example 17 pixels, the form and the grid change in steps of 17 pixels.

This is (part of) the code in our form's resize event:

```
* grid hovering and autoscrolling settings according to record height in grid
IF llHover = .T.
      LOCAL lcGridName, lnGridSpace, lnFormSpace, lnGridHeight
      lcGridName   = ThisForm.HoverContainer1.gridname
      lnFormSpace  = ThisForm.Height - ThisForm.&lcGridName..Height
      lnGridSpace  = FLOOR((ThisForm.&lcGridName..Height -
      ThisForm.&lcGridName..HeaderHeight - 2) / ThisForm.&lcGridName..RowHeight)
      lnGridHeight = 2 + (lnGridSpace * ThisForm.&lcGridName..RowHeight) +
      ThisForm.&lcGridName..HeaderHeight
      ThisForm.&lcGridName..Height = lnGridHeight
      ThisForm.Height = lnGridHeight + lnFormSpace
      ThisForm.HoverContainer1.Resize()
ENDIF
```

As can be seen this code makes use of the HoverContainer class. Our sample form will show the logic for this kind of form (and grid) resizing. We are very pleased with the results.

Sample form

The sample form, and the start.prg will show how to implement the HoverContainer class. Everything is there, including the grid sorting showing the small black triangles to show the data with ascending or descending order. The KeyboardEnter procedure is placed in the start.prg. The program start.prg will call the form with the required NAME clause.

To simplify the demonstration I have created a single field cursor in the form's LOAD. The default index is also created here.

In this sample a few variables are made public. This is only done here to make things not more complicated than they might already be. In real life these values are taken from a "setup" table. If you don't want any public variables you can change them into form properties or whatever you like and choose.

Click on the grid and see that hovering stops. Right click on the "SELECT" button and hovering will start again.

In the demo you will find a lot of comment lines describing the process step by step.

Release notes

This class has been developed in 2023 using VFP version 9, with Service Pack 2 installed. The exact build is 09.00.0000.7423. This class has not been tested with any other version of VFP (including VFPA). I do not know if any commands were used that are not available in other versions.

If you try this class with any other VFP version, please let me know the results, so I can update this document.

Last but not least…

Information, documentation and downloads: *www.winvis.nl/hovercontainer.html*

Contact: info AT winvis.nl or visit foxite.com

Gerrit J. Broekhuis
October 2023