Designing a Tabbed GUI Interface in RackAFX

Will Pirkle

The most often requested feature for the RackAFX GUI Designer is the ability to make “tabbed” interfaces where the selection of a tab, button or switch changes the contents of a view container. This advanced GUI control saves space on your GUI and provides an intuitive interface for complex plug-ins. RackAFX v6.6 adds this new feature. 27

In VSTGUI4, a tabbed GUI control consists of two components:

- a Tab Controller
- a UIViewSwitchContainer

Let’s look at some example plug-in interfaces that use this kind of control. First, an Izotope plug-in (NOTE: this is only for a visual example, it was not designed with RackAFX)

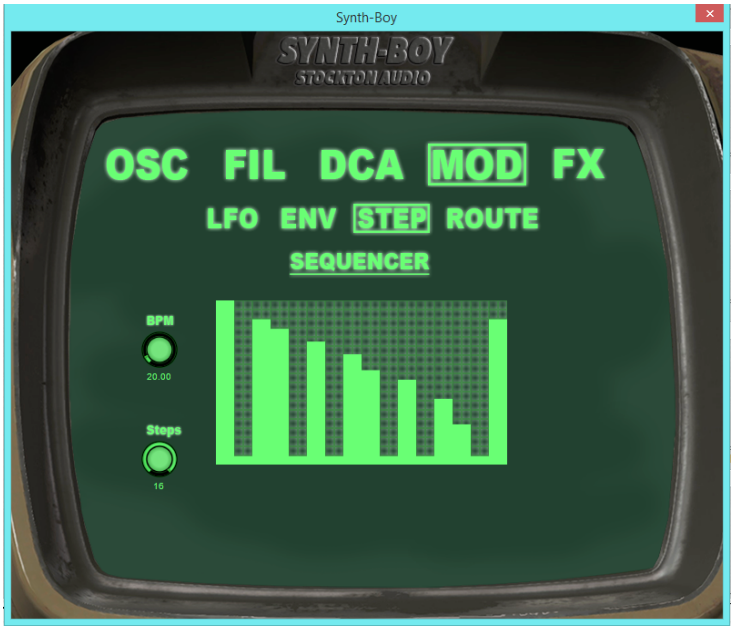
In this example, the Tab Controller does resemble file folder “tabs” and it switches the view content of the box just below it.

In this Apple AU plug-in, the Tab Controller is really a series of horizontal buttons, but the concept is the same:

Finally, the MiniSynth plug-in example also uses what appears to be a series of buttons (they are actually a new kind of control in v6.6 called the CHorizontalSwitch). This example shows the third tab selected as well.

Here is an excellent example of a tabbed GUI made in RackAFX from Andy Stockton. It is a synth GUI with multiple tabbed GUI elements:

The “OSC FIL DCA MOD FX” graphic is the tab controller. The MOD page shows an example of a tabbed GUI element inside of another View Switch Container:



Here, the “LFO ENV STEP FX” graphic is the 2nd tab controller embedded in the main outer tabbed GUI View Switch Controller.

A ***Tab Controller*** is a VSTGUI4 object which controls the selection of the current view. There are three possible Tab Controller objects:

1. CVerticalSwitch (new in RackAFX v6.6)
2. CHorizontalSwitch (new in RackAFX v6.6)
3. COptionMenu (aka Drop List)

These three objects are able to transmit zero-indexed integer values which in turn select the current view. The other VSTGUI4 objects transmit normalized values [0.0,1.0] by default and are generally not used as Tab Controllers (though you may certainly use the new RackAFX Custom View feature to create your own).

The CVerticalSwitch and CHorizontalSwitch are the preferred Tab Controllers. The COptionMenu can be used but requires some specialized setup in RackAFX (explained later in this document).

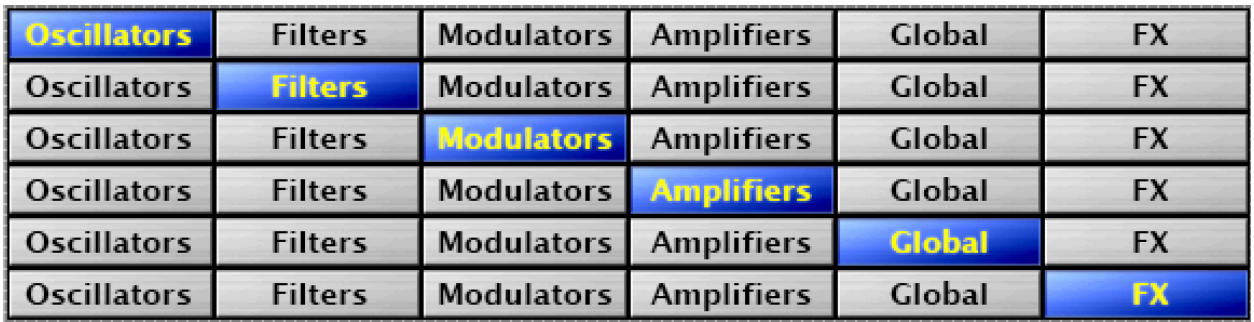
A ***UIViewSwitchContainer*** is a special kind of CViewContainer that shows and hides a series of sub-views, all of which must be CViewContainers. The UIViewSwitchContainer shows and hides the subviews with an animation that fades the old CViewContainer out as the new CViewContainer fades into view, making it a slick and professional looking control. You can control the animation time for the view fade in/out.

Some simple rules about using the Tab Controller/UIViewSwitchContainer:

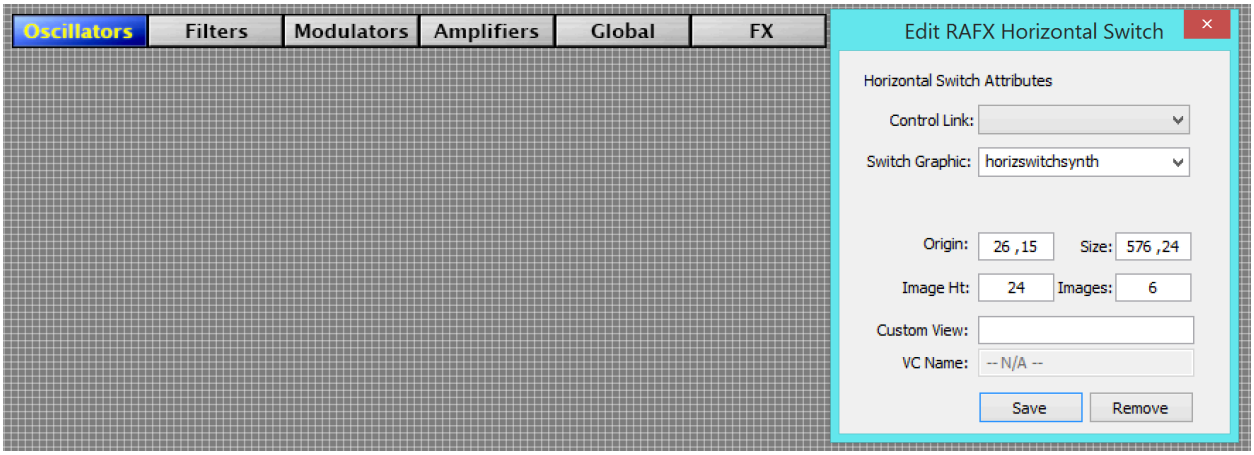
1. The UIViewSwitchContainer can only show/hide CViewContainers
2. The Tab Controller must have the same number of index values as the UIViewSwitchContainer has sub-views — if you get this wrong, the Tab Controller will either skip over views, or select the wrong one.
3. The UIViewSwitchContainer can have a background color, but NOT a background graphic image. It can also be transparent.
4. In general, you want your sub-view CViewContainers to have the exact same dimensions (width, height) as the UIViewSwitchContainer.
5. The sub-view CViewContainers will be automatically placed in the UIViewSwitchContainer with the upper left corners (0,0) matching up.
6. You can have as many UIViewSwitchContainers as you want, and you can also embed them inside one another.

CHorizontalSwitch
Ordinarily, this control is used to make customized Radio Buttons which are displayed horizontally, and you can use it for that as well.

This is the most common kind of Tab Controller. By its own nature, it is also a very customized control and will require you to generate your own graphics. RackAFX ships with one built-in CHorizontalSwitch graphic that is shown in the MiniSynth example above. The graphics you use can be as complex as you want. In the MiniSynth example, you see only one row of items (here they resemble buttons, but they do not have to — since the graphic is custom, you can create whatever you want such as tabs, icons, etc…). However, the complete graphic required shows all buttons in all states, one row at a time:



I used KnobMan to make the two button states (highlighted or not) and then assembled the final graphic in PhotoShop, using a bit of the outer glow effect to make the yellow text light up. Here is the CHorizontalSwitch added to the RackAFX GUI Designer:



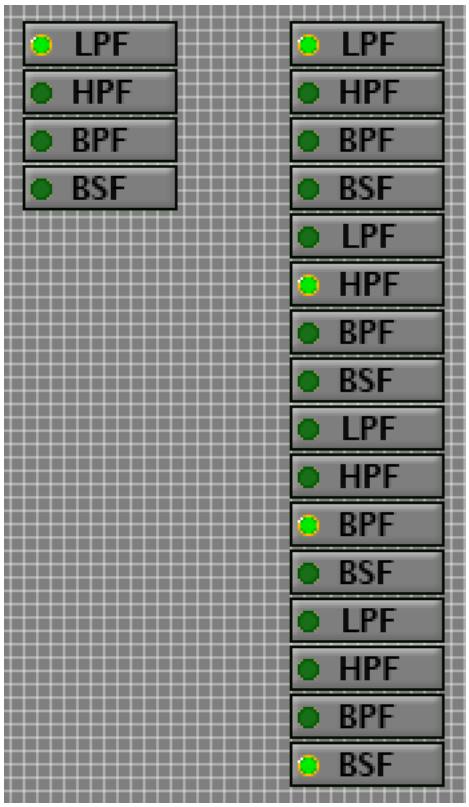
- The size (576,24) is the (width, height) of one row of items, not the entire graphic.
- Image Height is the height of one row (24 pixels)
- The Images value is the number of discrete items (tabs, buttons or whatever you are using) that will be switched, and is also the number of rows in the complete graphic. This value must be set properly for the control to function correctly.

CVerticalSwitch

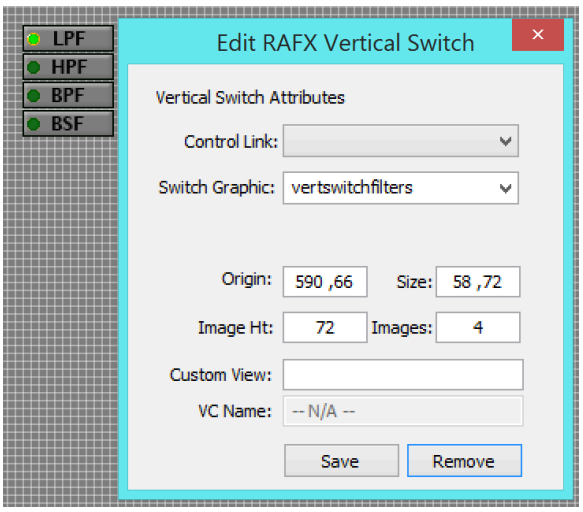
Ordinarily, this control is used to make customized Radio Buttons which are displayed vertically, and you can use it for that as well. In fact, this control was originally omitted from RackAFX v6.5 because it requires you to generate your own graphic, unless you happen to need the one graphic that ships with RackAFX, below.

This is the second most common kind of Tab Controller. Like the horizontal version, it is also a very customized control and will require you to generate your own graphics. RackAFX ships with one built-in CVerticalSwitch graphic that is show here.

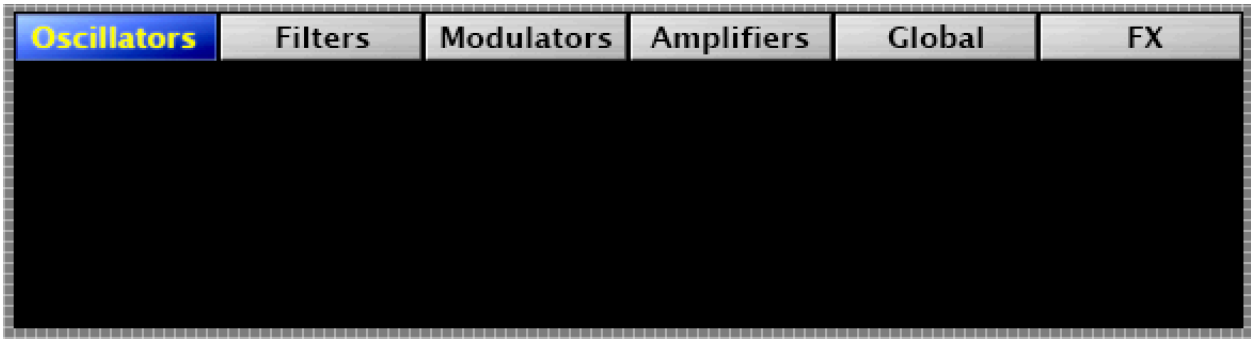
On the left is what the switch looks like on the final GUI, and on the right you can see the complete graphic - like the CHorizontalSwitch, it consists of a series of sets of buttons which show the control in every possible state. The sets of buttons are stacked vertically, each showing one selected item.



The same set of rules apply regarding size, image height and image count values.

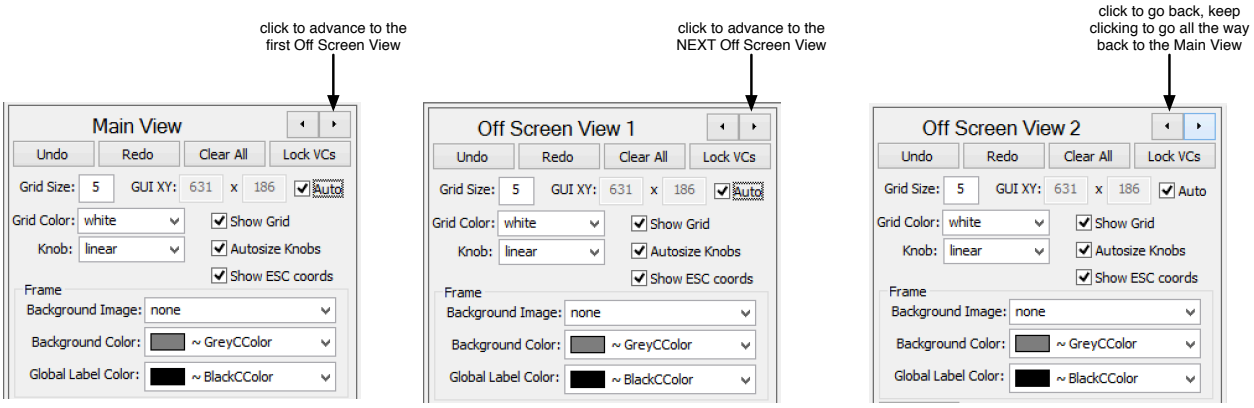


UIViewSwitchContainer and Off Screen Views
When you drag the UIViewSwitchContainer control into the RackAFX GUI Designer, you will only see a black box, that you can resize to the proper dimensions for your own GUI. This is what it looks like for the MiniSynth example:



On its own, the UIViewSwitchContainer doesn't look like much. We need to generate some CViewContainers to populate it and give it some views to switch. The problem is that in the old RackAFX GUI Designer, anything you drag and drop into the designer will show up on the GUI. We need a way to easily generate these CViewContainers with the same drag and drop interface, but we need them to not be visible on the main GUI. To do this, I used a paradigm called "off screen views" which is loosely based on the same concept in VSTGUI4 (or any GUI library for that matter). In this paradigm, you synthesize views off the screen (hidden) then fly them in as needed.

In my version, these off screen views are design areas that you switch between using the buttons at the top left of the GUI Designer — you can have as many off screen views as you want.



Use the nudge buttons to advance forward or backwards through the list of Off Screen Views. The Main View is always the first view in the sequence.

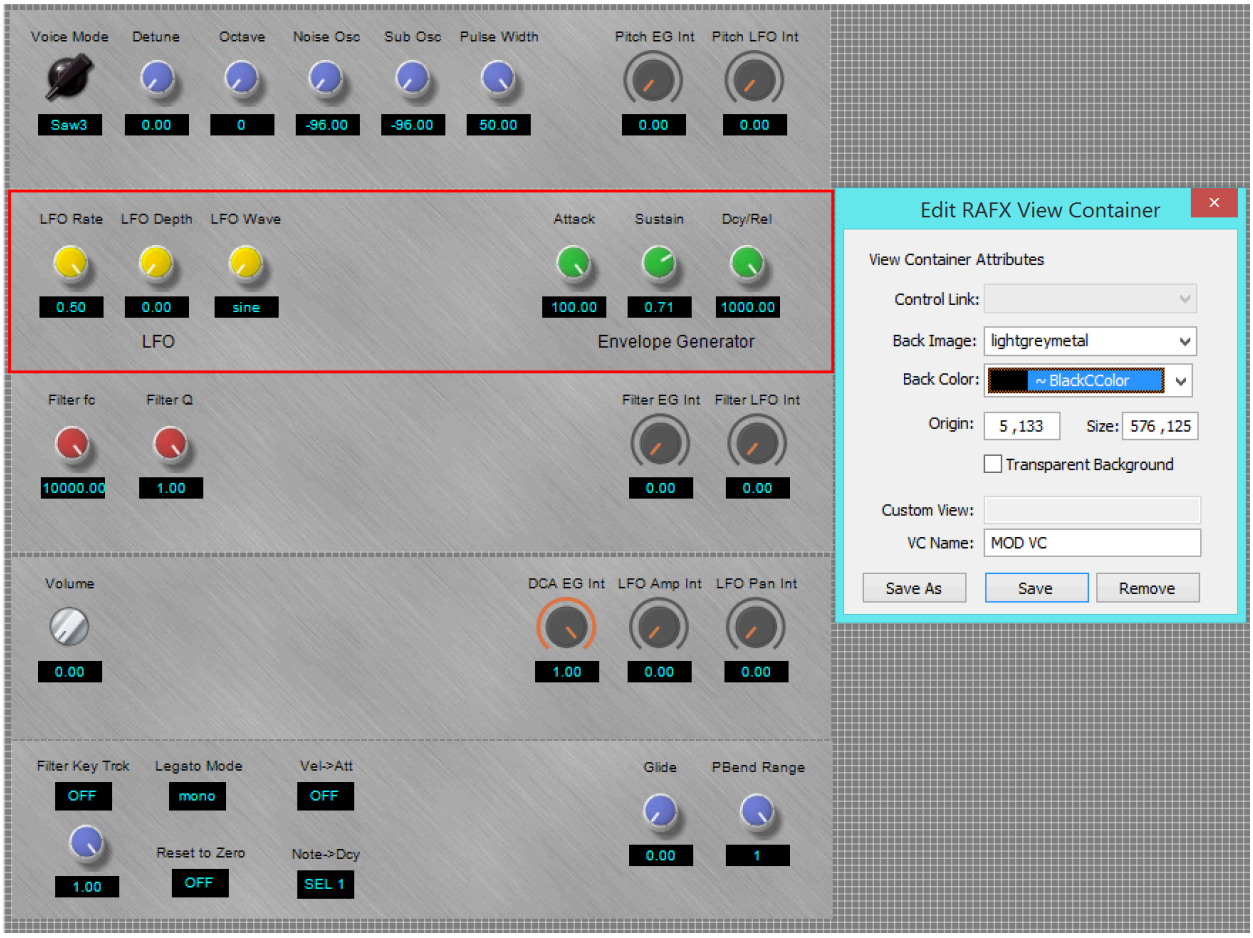
The Off Screen Views are there specifically for you to design the CViewContainers for your Tabbed GUI control. Lets take a look at the first CViewContainer I designed for the MiniSynth GUI. This image is from Off Screen View 1 in RackAFX:

Here, I dragged and dropped a CViewContainer and gave it the exact dimensions of the UIViewSwitch-Container (576,125) and gave it a Transparent Background, then I dragged the knob controls into it and lined everything up using the grid. Finally, I removed the transparency and added the Back Image of light-greymetal.

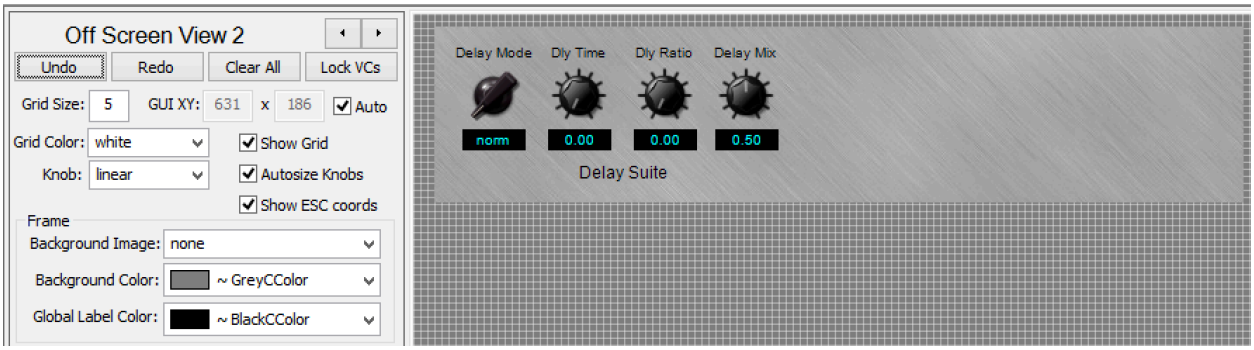
However, the most important concept in this example is the "VC Name" element.

> In VSTGUI4, all CViewContainers are referenced with **unique names**. RackAFX will generate these unique names for you, however you will probably want to change these names to something more specific and appropriate. In this example, I have changed the name to **OSC VC** and it is important to remember that if you want to use your own VC names, you must make sure they are unique!

When you assign the CViewContainers to the UIViewSwitchContainer, you will use the VC Names to do so (VSTGUI4 code also uses the names to connect the views to the switch-container). Here is what my Off Screen View 1 looks like in its entirety - it contains five of the six CViewContainers I need for the MiniSynth GUI:



Notice that I've selected the second container to show its unique name **MOD VC**. The sixth CViewContainer is assembled on Off Screen View 2 and contains the delay/FX controls and the VC Name is **FX VC**.



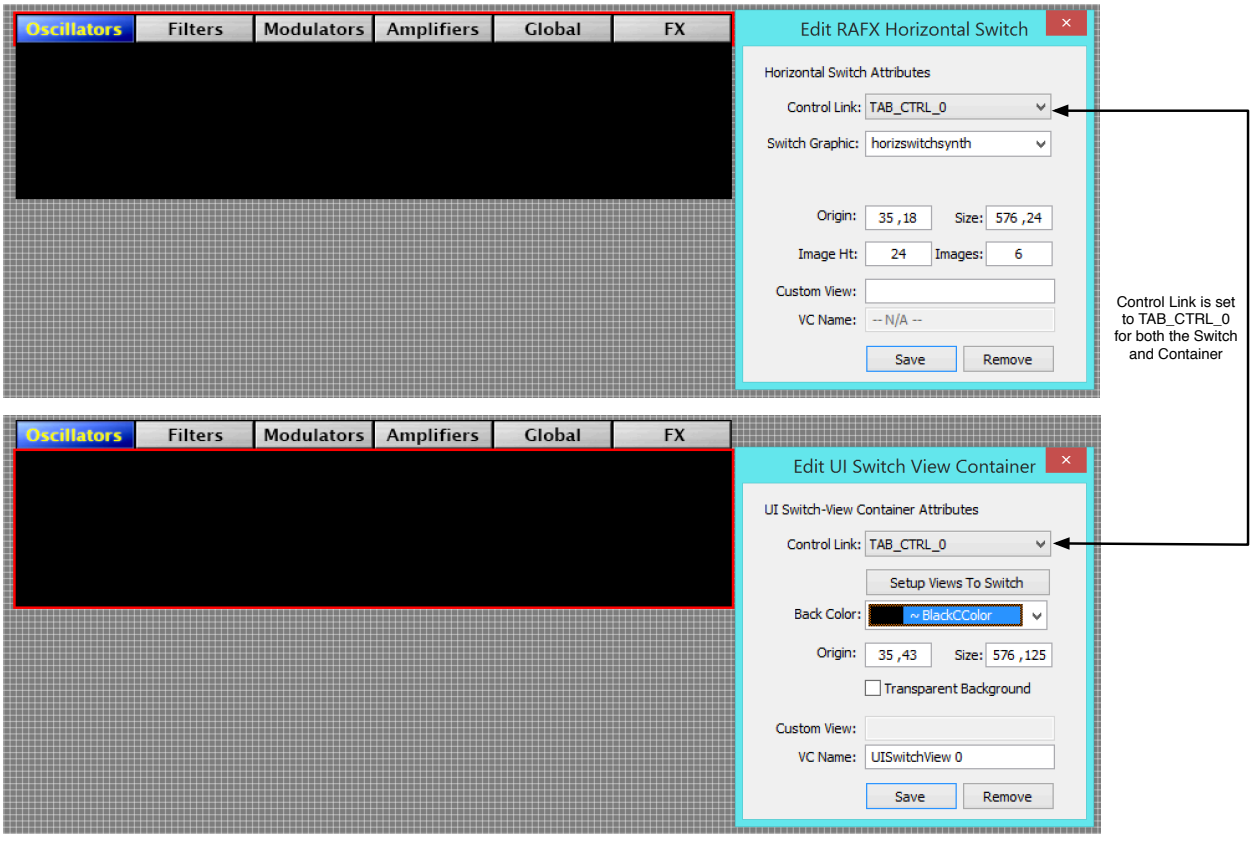
When using the Off Screen Views there are a few things to note/remember:

- It doesn't matter where you place the CViewContainers on the Off Screen Views — they will always be placed in the upper left of the UIViewSwitchContainer. I lined mine up neatly but you do not have to.
- The Undo/Redo buttons will also work for Off Screen Views. The Undo/Redo cache is cleared and reset each time you switch to/from the main view or any off screen views.
- You can not cut and paste from one Off Screen View to another, or to the Main View however you can always use the "User Templates" concept to create your own templates that can easily be added to any of the views.
- The Background Color/Image for Off Screen Views will not be seen — only the color/image of the Main View will appear on the GUI.

Completing the GUI by connecting the Tab Controller and CViewContainers
With the CViewContainers designed and ready on the Off Screen Views, there are only two steps left to complete the Tabbed GUI - connecting the Tab Controller to the UIViewSwitchContainer and setting the Views to switch between.

Control Link
You link the Tab Controller to the UIViewSwitchContainer via the Control Link. For all the other VSTGUI objects, the Control Link is the index value of the GUI control that you connect in the GUI Designer. This value is then used to connect your GUI element up to the underlying RackAFX variable which changes automatically as the user moves the control.

However, in the case of the Tab Controller/UIViewSwitchContainer paradigm, the Control Link really does not need to connect to a RackAFX variable (unless you want it to, explained below). So, I included 32 pre-set Tab Controller Control Links named TAB_CTRL_0, TAB_CTRL_1, TAB_CTRL_2,…,TAB_CTRL_31. These map to Control ID values 65536, 65537,… This allows you to easily implement up to 32 separate sets of Tab Controller/UIViewSwitchContainers. Right click on the Horizontal Switch and the View Switch Container and set their Control Links to the same item, here it is TAB_CTRL_0. This connects the two GUI elements together.

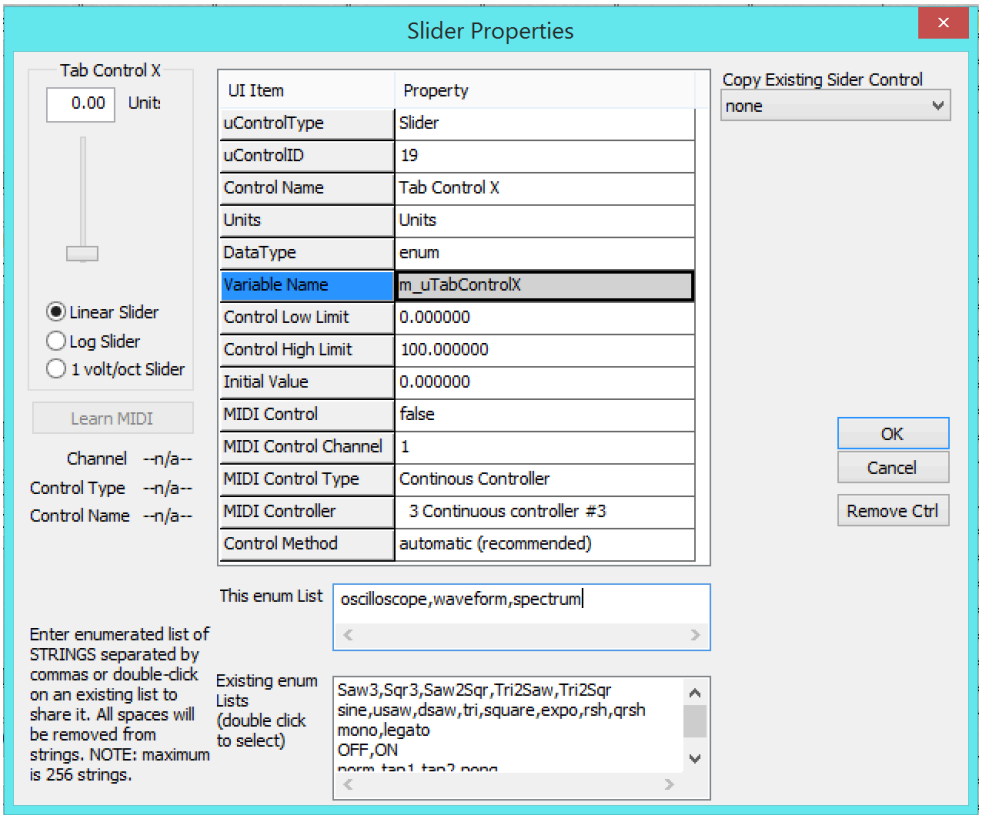


If you need more than 32 tabbed view-switchers in your GUI, you can manually add more Control Links. You can also do this to have the Tab Controller connect to an underlying variable in RackAFX so that you will get notified when the views switch.

If you want to use the COptionMenu (Drop List control) as a Tab Controller, you need to use this same custom control link setup.

To setup a custom Tab Controller Control Link, just create a "dummy" GUI control in RackAFX and set it to the UINT data type. Then, add as many control strings as you have views to switch between. For example, suppose you have a special Tabbed GUI control that shows audio data in three modes: oscilloscope, waveform and spectrum (your tab control would then need three image sets).

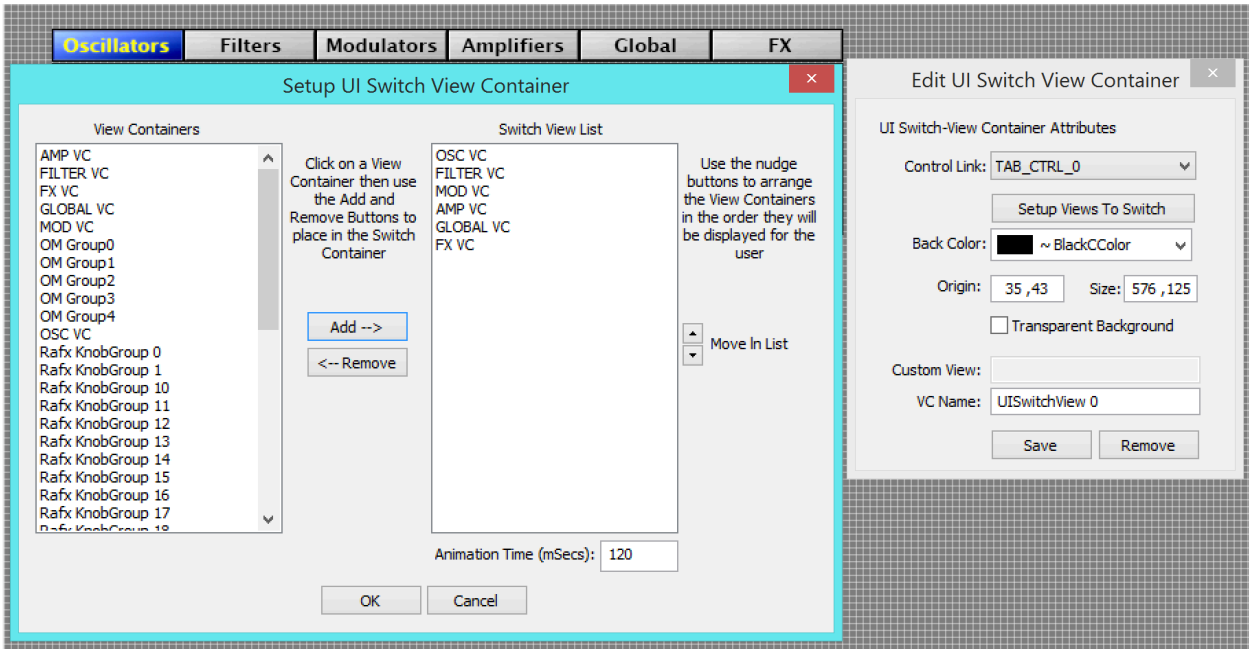
Here, I have create a control named ***Tab Control X*** and given it 3 strings in the enum list to select the three views. Then, in the GUI Designer I would select "Tab Control X" as the Control Link for both the CHorizontalSwitch and the UIViewSwitchContainer. In userInterfaceChange() you would trap the Control ID to get notifications about the view being switched (here it is Control ID = 19 as shown in the second item in the list).



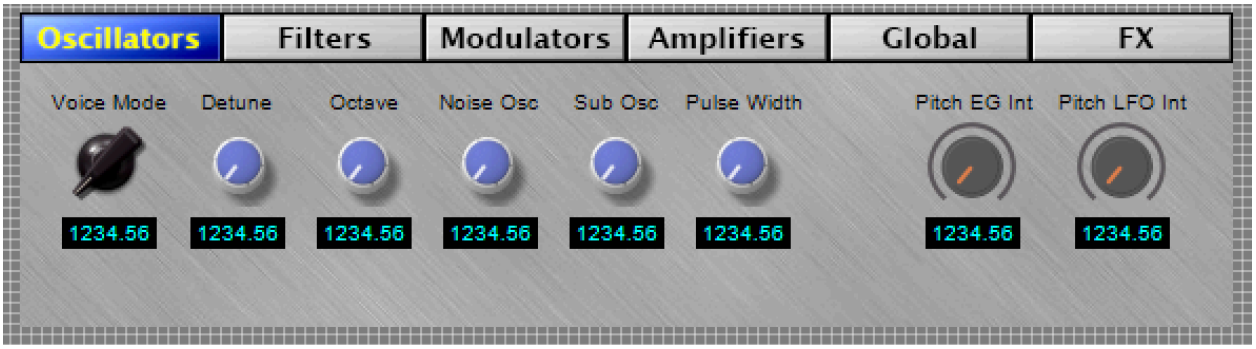
If you are using the COptionMenu as a Tab Controller, the enum strings will populate your Option Menu for selecting the appropriate view.

Setting up the Views to Switch

The last thing you need to do is setup the Views to switch between. This is done by clicking the button "Setup Views To Switch" in the UIViewSwitchContainer's setup dialog. When you click this, a dialog box will pop up with every CViewContainer in your GUI listed on the left. You add, remove, and rearrange CViewContainers with the appropriate buttons. Here, I've added the six different CViewContainers with my own custom VC Names and arranged them to match the Tab Control button order. Hint - you can also double click an item in the View Containers list to automatically add it to the Switch View List.



Note the ordering of the views in the Switch View List on the right. Also notice that here is where you can set the Animation Time in milliseconds (the fade in/out time). Setting this variable to 0 will instantly switch views with no fade in/out. After you finish setting up the list of views to switch, you will notice that the GUI Designer will now display whatever View Container you have at the top of the list inside the black box:



This is purely "for show only" to let you see the first of the views that will be switched. You can not edit or move this first view. Right-clicking on it will simply bring up the UIViewSwitchContainer's setup dialog.

Compile and Go!
Finally, exit the GUI Designer and recompile your plug-in. The new Tabbed GUI interface, as with all RackAFX GUIs, is platform independent and will show up identically in RackAFX, VST and AU.

References:

VSTGUI4 Files and Documentation: http://sourceforge.net/projects/vstgui/