PentaPanes Class Demo

Description:

This example *Penta Panes Demo.vi* is a **mockup** graphical user interface (GUI) window. it demonstrates the use of the "PentaPanes" class (contained within *Penta Panes.lvlibp*).

The PentaPanes class was written to fulfill the following GUI window requirements

- GUI window has to fit and be functional across multiple **screen resolutions**.
- GUI window has **re-sizable.**
- Certain GUI elements (like graphs) need to change size with the front panel, while other elements need to keep appropriate size and position.
- GUI window has to be able to accommodate many actions, settings, parameters, calibration procedures, measurements, etc. all of which needs to be easily accessible while keeping the Front Panel **uncluttered** (so as to not intimidate users with a messy confusing interface), with a logical meaning (e.g. grouped by functionality) and is not wasteful of precious front panel area which needs to be used for viewing the actual work data.
- GUI needs to be able to contain hidden and/or restricted areas accessible only to developers (e.g. debugging indicators, advanced users or accessing factory settings).
- GUI needs to to be able to contain **side by side** view of **several graphs**, possibly of different types, for comparisons and multi-dimensional data viewing, line profiling etc.
- The GUI window will pose no restriction on the UI/UX designer – any type of interface can be incorporated into the design. Menu, Icons, Right mouse clicks
- Intensity graph needs to be able to be resized while keeping the **aspect ratio** of the axes.

What this class does is handle the setup GUI window (a FrontPanel window composed of 5 panes created by 4 splitters) automatically. Giving the developer methods to open, close and resize panes, determine how each GUI element will behave when its pane is being resized. It also allows you to have multiple panes "instances" by placing them outside the visible area of the pane and then switching the pane origin to show them - using *SetPaneOriginToPaneInstance.v*i .

The purpose of using this class (or this demo as a template) is to have a GUI window with all the mechanisms of resizing and panes-works already built-in, so the developer can go straight to working on the actual functionality of the software.

a few notes:

- The LabVIEW class (*5PanesGUI.lvclass*) is contained within the packed library *Penta Panes.lvlibp* It is the compiled code of the class with the diagrams removed so at the present time only using the class is possible, not editing the class code.
- To view information of this class’s methods (as well as other VIs) use the “Context help” window.
- currently only this configuration of splitters is allowed (two horizontal and then two vertical in middle pane).
- splitters can be any width however LabVIEW doesn't allow changing splitter width during runtime.
- currently pane's scrollbars aren't allowed and should be set to invisible. In registered objects scrollbars are OK.

I hope many of you will find this example useful and effort saving.

How to use:

Running the Demo

1. Download/Extract the files into a directory
2. Open project: *PentaPanes Demo.lvproj (*on LabVIEW 2023 or compatible*)*
3. Run topVI: *Penta Panes Demo.vi*

In order to use the class in your GUI

1. Build an appropriate FP (two horizontal splitters and then in the middle pane two vertical splitters)
2. Init the object using *Init5Panes.vi*
3. Register controls and indicators objects using *RegisterGUIObjectToResize.vi*
   Note: currently the supported objects for use in this class are
   SubPanel, Path, String, Table, TreeControl, Array, GraphChart (and all child classes), TabControl, Boolean, Picture.
4. Automatically resize all objects in pane by adding *ResizeObjectsinPane.vi* to **Pane Size** events.
5. Resize panes by using *SetPaneSize.vi* to close pane resize it to 0.
6. If you want to change the layout during runtime you may re-register objects (using *RegisterGUIObjectToResize.vi* with the same reference again).
7. Shutdown the object before the FP closes using *ShutDown5Panes.vi*

Additional Information:

SetWindowIcon.vi
This demo uses the *SetWindowIcon.VI* to change the default LabVIEW window icon (top left corner of the window during runtime) to a dragonfly.
I didn't write this VI, I got the code snippet from NI community.
https://forums.ni.com/t5/LabVIEW/Changing-Icon-in-LabVIEW-Title-bar-of-a-vi/m-p/1480986#M560223

Note: This VI is the only part of the code that is Windows specific, if you are running this on a non-windows target you should remove this VI and everything should work (not tested).

DeferPaneUpdate
In order to make the demo run more smoothly at the start of the Init phase sequence the FP state is set to hidden, and Panel's DeferPanelUpdate is set to True. DeferPanelUpdate is also set to True before running each event loop
DeferPanelUpdate is set to False on eventloop timeout (and on the first call also changes FP to visible).
However if you want FrontPanel to update **during the event** you can set DeferPanelUpdate to False at the beginning of the event. (as demonstrated in the event "Icons" Mouse Down?)

Icons line
Since I wanted to demonstrate how this class handles image arrays I used an **image array as icons** buttons

You can review in the code how to populate the image array with icon images from a directory using *LoadIconsIntoArray.vi*
You can check out the Icons: Mouse Down event which figures out which icons was pressed and by what mouse button and acts accordingly. Icons can be used as "Latch action" or "Switch action" buttons.
Note: In this demo only two icons actually do something,

icon #18 "log" button is a switch action button that shows and hide the bottom pane (containing a log)
icon #9 "zoom-to-actual-size" button is a latch action button that auto-scales all visible graphs