PentaPanes Class Demo

Description:

This example (Top VI *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 requirements:

- GUI window has to fit and be functional across multiple **screen resolutions**.
- GUI window has to be **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.
- The GUI should have a possibility to make it **uncluttered** (so as to not intimidate users with a messy confusing interface) and it should be easy to place controls according to logical meaning (e.g. grouped by functionality).
- The interface itself should not be wasteful of precious panel area (which should be utilized 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 interface 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, multi-dimensional data viewing and actions like line profiling.
- The GUI window will pose no limitations on the **UI/UX designer freedom**, any type of interface can be incorporated into the design like object's label and scroll-bars, images, online menus, icons etc.
- Intensity graph should to be able keep their axes' **aspect ratio** when resized.
- GUI building shouldn't require placing the controls (or indicators) in a pixel accurate position.

What this class actually does is handle the setup of the interface window (i.e. FrontPanel window composed of **5 panes** created by **4 splitters**) automatically. Giving the developer **methods to open/close/resize** panes and determine how each GUI element will behave when it is being resized. It also contain methods to **resize a specific object, all object's in a pane or all objects in all panes.** It also allows you to have multiple pane "instances" (by defining object outside the visible area of the pane and then switching the pane origin to show them, using *SetPaneOriginToPaneInstance.v*i) thus giving the developer virtually unlimited user interface space.

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.

Notes:

- To view information of this class's methods (as well as other VIs) use the "**Context help**" window.
- The LabVIEW class (*5PanesGUI.lvclass*) is contained within the packed library *Penta Panes.lvlibp* It is the compiled code with the diagrams removed so at the present time there is no access to the internal class code.
- 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:

<u>Running the Demo</u>

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*

<u>In order to use the class in your GUI</u>

1. Build an appropriate FP (two horizontal splitters and then in the middle pane two vertical splitters)
2. Place controls and indicator in their corresponding panes.
3. Init the object using *Init5Panes.vi*
4. Register controls and indicators objects you placed in step 2 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.
5. Automatically resize all objects in pane by adding *ResizeObjectsinPane.vi* to "Pane Size events".
6. Resize panes by using *SetPaneSize.vi (*to close pane resize it to 0).
7. If you want to change the layout during runtime you may re-register objects (using *RegisterGUIObjectToResize.vi* with the same reference again).
8. Shutdown the object before the FP closes using *ShutDown5Panes.vi*

Additional Information:

This is information regarding the demo program itself and is unrelated to the PentaPanes Class.

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 each event loop case, so if you want to include an event that needs to change the panel **during** the event make sure to set DeferPaneUpdate to False inside the event before running the code that changes the panel (see event “Icons: Mouse Down?” For an example). In any case, DeferPanelUpdate is set to False on the **timeout** frame of the event structure (along with changing the FP to visible if it is the first call to the timeout), So normally panel only updates when event loop timeouts.

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

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 at:

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).