LABVIEW™ TECHNICAL RESOURCE

THE ONLY LABVIEW SUBSCRIPTION WITH VI SOFTWARE INCLUDED

Volume 7, Number 3 $95 yr/U.S.

THE LEADING INDEPENDENT TECHNICAL RESOURCE FOR LABVIEW USERS

On This Issue's Disk!

Custom Control Template...
Table Print Utility...
VI Search/Report Tools...

A publication of LTR Publishing, Inc.

LabVIEW Special Effects

GUI Techniques Using Attribute Nodes

by Dave Ritter

With the introduction of ActiveX in LabVIEW 5, and now with the inclusion of the Picture Control Toolkit in LabVIEW 5.1, there are a multitude of slick interface options and GUI enhancements just waiting to be plugged into your latest LabVIEW application. But hold on a second, before we get too carried away with all of the great new toys, let's not forget that venerable custom interface workhorse: the humble attribute node.

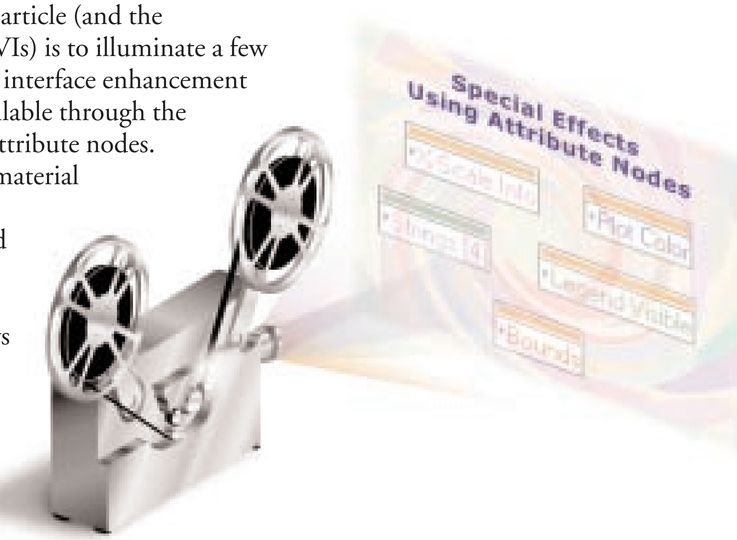
LabVIEW GUI Techniques on this Issue's Disk:

Simulated Mouse-over Effect...
Drag&Drop Effect...
Scrolling Text Effect...
"Fade to Black" Effect...

Attribute nodes are often the quickest and easiest way to alter your user interface "on the fly", and what's more, they are "100% Pure G". For the Windows-based LabVIEW developer, this means that if you decide (or need) to port your application to LabVIEW for Linux, MacOS or Unix at some point down the road, you will have spared yourself the unpleasant task of replacing all of those platform-specific ActiveX controls.

Long before the Picture Control Toolkit and ActiveX were supplied as standard parts of the LabVIEW development system, savvy LabVIEWers were finding clever ways to solve GUI challenges simply by harnessing the power of attribute nodes. In fact, many "cool" and elegant user interface designs rely heavily on clever attribute node tricks to extend the functionality of standard LabVIEW interface elements.

The aim of this article (and the accompanying VIs) is to illuminate a few of the countless interface enhancement possibilities available through the creative use of attribute nodes. Hopefully this material will spark your imagination and help you find a few simple and elegant ways to improve your own VIs using similar attribute node tricks.



continues on page 4

LabVIEW Special Effects
continued from page 1

Whether you currently use attribute nodes to dynamically build user-configured HMIs at runtime, or generally chose to ignore attribute nodes altogether, there is probably a good chance that you will find something here to set you on the path to enhanced attribute node discovery.

Special Effects Examples Using Attribute Nodes

The following attribute node demo VIs have been included on this issue's LTR resource disk:

- Simulated JavaScript Mouse-over Effect
- LabVIEW Drag & Drop for user defined interfaces
- Scrolling Text About Box
- "Fade to Black" Text Effect

SPECIAL EFFECT EXAMPLE 1: Simulated JavaScript Mouse-over Effect

[*see* mouseover.vi *in the* Attribute Nodes.llb *file on the LTR resource disk*]

If you have had the opportunity to spend some time on the Internet surfing the web, this particular GUI effect may be quite familiar to you. First appearing as a popular interface enhancement for interactive media and CD-ROM titles, the "mouse-over" effect was later adapted to JavaScript for web use, and has since become a favorite of countless web page designers and programmers. Quite simply, when you pass your mouse over a predefined "hot spot" on a suitably enabled web page, some part of the screen image changes in response to the mouse position. Moving the mouse pointer away from the hot spot causes the altered image to revert to the original state. Often, several hot spots are programmed into a single web page, each triggering a different change to the screen image. Frequently, navigation buttons are given this treatment; to attract the eye and provide some visual interest to an otherwise boring and static web page.

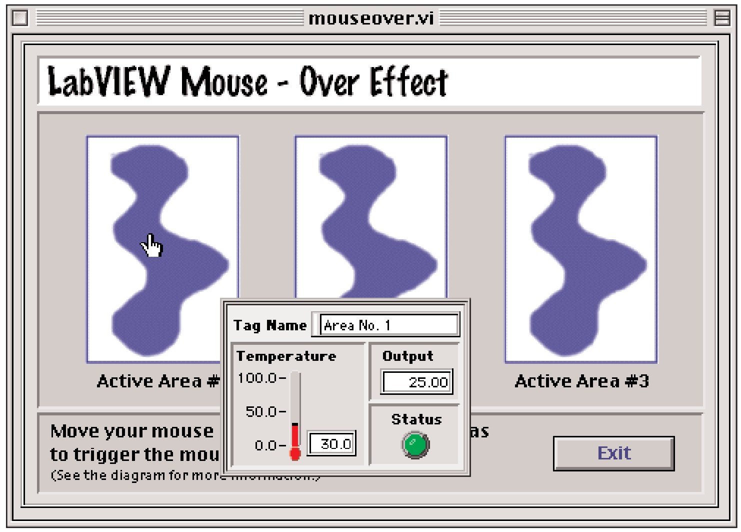


Figure 1a: As the mouse passes over one of three predefined "hot spots", a corresponding display cluster pops up to reveal hidden parameter values.

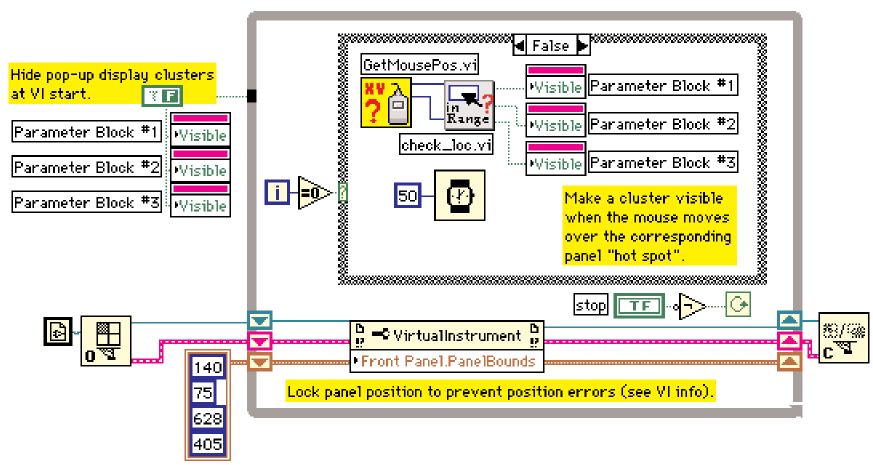


Figure 1b: The current mouse location is compared with the coordinates of predefined hot-spots inside check_loc.vi. If a mouse-over state is detected, the corresponding cluster is revealed by setting its "visible" attribute to TRUE. Also notice that VI Server is used to prevent panel relocation at runtime.

This technique can be easily adapted for the LabVIEW environment to provide a variety of interesting GUI enhancement possibilities. A few that immediately come to mind include:

1) toggle the state of a boolean indicator when the mouse passes over;
2) advance to the next frame of a pict ring indicator;
3) change the color of a text box;
4) reveal a hidden control, indicator, graph, etc.

These and several other applications of the mouse-over effect could prove quite useful given the appropriate situation, but our sample VI will employ item #4 from the list to address a specific LabVIEW programming challenge – reducing front panel clutter.

If you are like many other LabVIEW programmers, you probably find that, as the complexity of your application grows, so too grows the number of front panel controls, indicators and display items (not to mention the headache of keeping your application user-friendly and aesthetically pleasing). You may have spent several hours shifting dozens of front panel objects around in a vain attempt to find a visually appealing combination. You may even have invested the time required to develop an elaborate GUI system that employs attribute nodes to hide and reveal front panel controls and indicators, all in an effort to keep clutter minimized and operator comprehension maximized. If any of this sounds a bit too familiar, or if you've just wondered if there is an easier way to minimize panel clutter without resorting to one of the more complicated (and time consuming) interface configurations, consider this solution:

> Use mouse-over effects to display indicators (or clusters of indicators) only when the mouse passes over panel "hot spots."

Figure 2a: In "standard mode", pressing one of the numbered buttons triggers the corresponding LED indicator.

Figure 2b: Enter "drag mode" by clicking on "Activate" in the Drag & Drop menu.

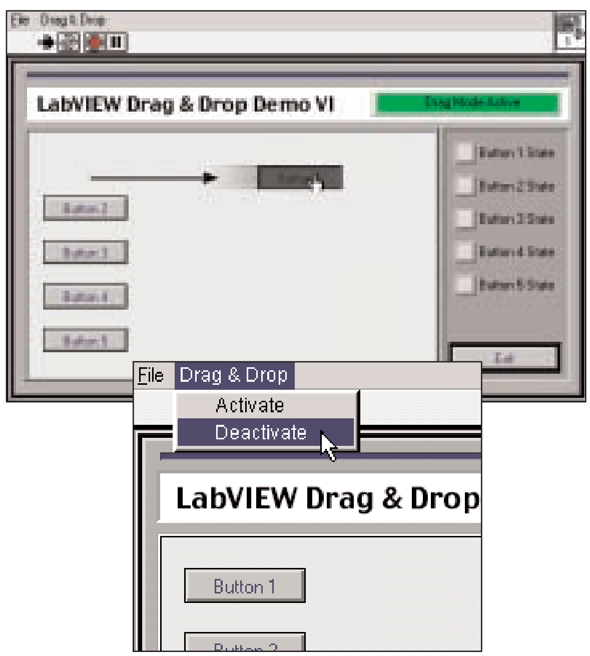


Figure 2c: Click and hold on a button until it turns gray (disabled), then drag the button to a new location on the panel. All five buttons can be moved freely and repeatedly while in "drag mode".

Figure 2d: DandD.vi Returning to "Standard Mode" – When the buttons have been moved to the desired locations, click "Deactivate" in the Drag & Drop menu to return to "standard" operating mode and restore regular button functionality.

The example mouseover.vi, shown in ***Figure 1a and 1b***, provides a very simple demonstration of this technique. While you are examining the sample VI, take a moment to consider the variety of ways this basic concept could be extended to add useful interface enhancements to your own projects.

How Does It Work?

1) The mouse position is polled using a platform-dependent mouse position VI (or alternatively, a mouse position

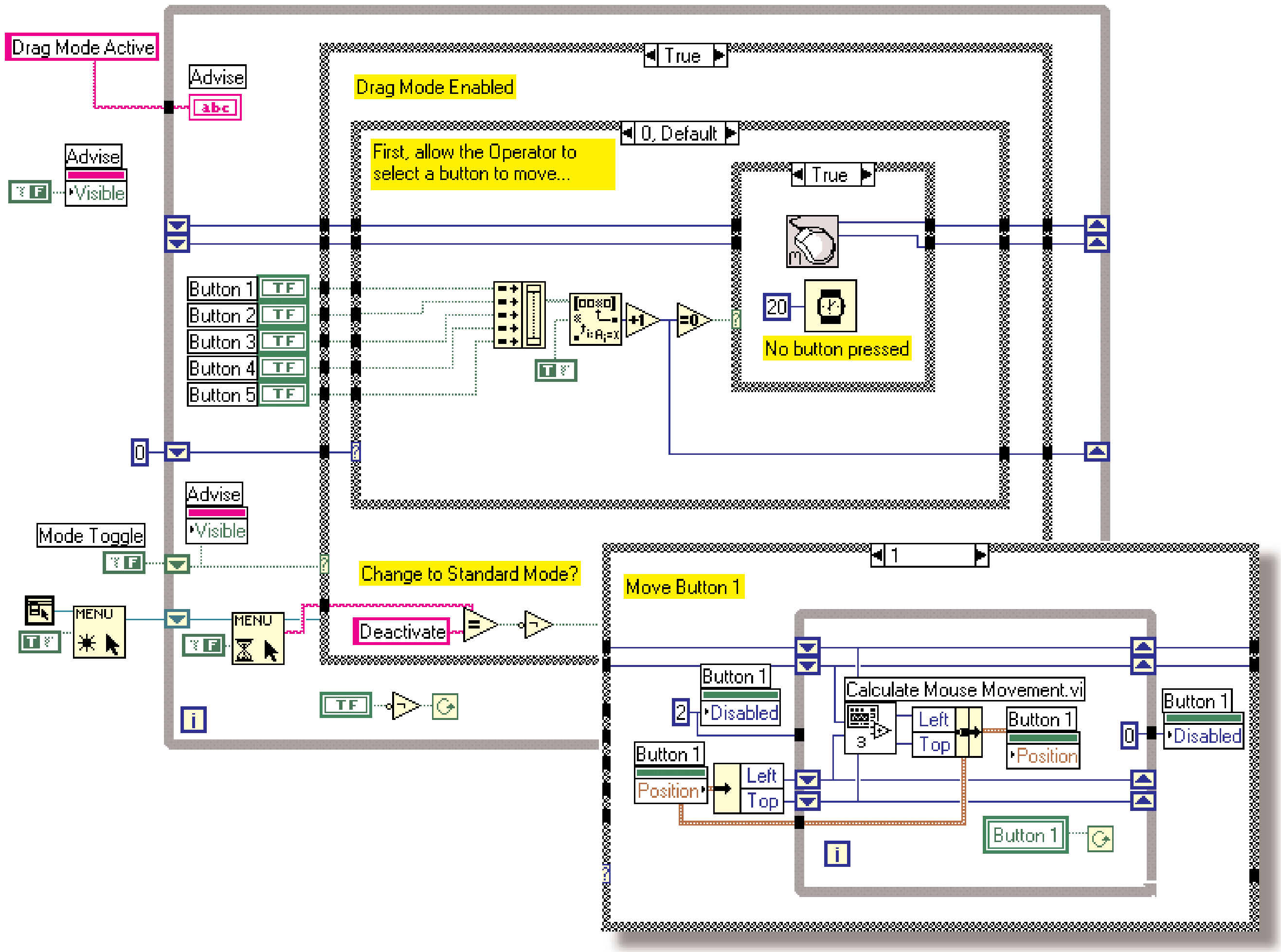


Figure 2e: After "drag mode" is activated from the Drag & Drop menu, the VI waits for one of the five buttons to be pressed (see 0, Default case above). When one of the five buttons is pressed, we proceed to the corresponding "button move" logic. The logic for button #1 is shown in the "case 1" inset.

attribute of a picture control – now a standard feature in LabVIEW 5.1);

2) The mouse position is compared with the location of the three predefined hot spots;
3) If a "collision" or "mouse-over" condition is detected, the corresponding Parameter Block cluster is made visible.
4) When the mouse moves out of the targeted hot spot range, the Parameter Block is hidden.

SPECIAL EFFECT EXAMPLE 2: LabVIEW Drag & Drop for user defined interfaces

[*see* DandD.vi *in the* Attribute Nodes.llb *file on the LTR resource disk*]

Another interesting application with attribute node potential is the implementation of runtime "drag & drop" functionality. While this feature is necessarily limited in range to a single front panel (in other words, no component dragging is possible between LabVIEW panels or to the desktop), it still provides some interesting possibilities for end-user defined GUIs, or drag & drop configuration screens.

The sample VI, **shown in** *Figure 2a–2e*, demonstrates a very simple implementation of an end-user configurable interface panel. In this example, the user can enter a special "drag mode" to rearrange the various buttons on the panel. After moving the buttons to the desired locations, the user can exit "drag mode" and resume normal program operations.

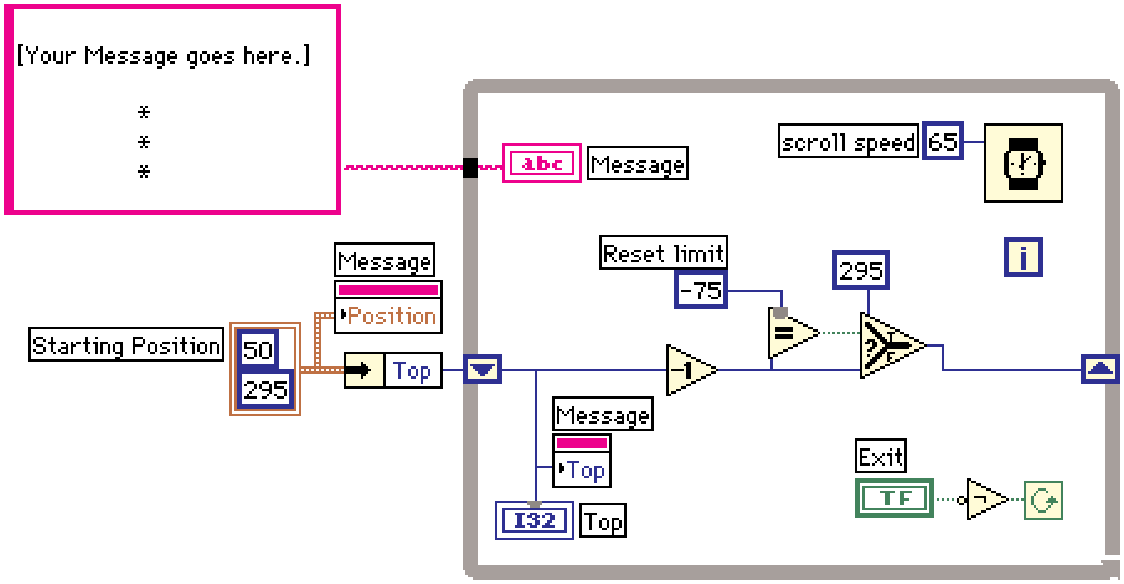


Figure 3: Repeatedly decrementing the "top" parameter of the message "position" attribute node scrolls the text up the screen one pixel at a time. When the reset limit is reached, "top" is restored to the initial value of 295 and the process repeats.

How Does It Work?

1) A custom "Drag & Drop" menu is created and polled for an "Activate" request. Until "drag mode" has been activated by selecting "Activate" in the "Drag & Drop" menu, the VI remains in "standard mode."
2) In "standard mode", five numbered boolean buttons are wired to five LED indicators. When any of the five buttons are pressed, the corresponding LED indicator will be triggered.
3) When "Activate" is selected from the "Drag & Drop" menu, the VI enters "drag mode". In drag mode, any of the five boolean buttons can be moved by clicking and holding the mouse button, then dragging the button to the desired location.

   Tip: Press and hold the mouse button until the boolean appears disabled (grayed out). When it appears disabled, it is ready to be moved to any location on the panel. Dragging a boolean button before it turns gray may cause a "jump" (rather than a "scroll") to the new position. This effect will likely be more pronounced on slower systems as the LabVIEW diagram must catch up to the new mouse position.
4) After moving buttons 1 through 5 to new preferred locations, select "Deactivate" from the "Drag & Drop" menu to return to standard mode. (*see item 2 in this section*)

SPECIAL EFFECT EXAMPLE 3: Vertical Text Scrolling About Box

[*see* scroll.vi *in the* Attribute Nodes.llb *file on the LTR resource disk*]

Scrolling text is often a great option for kiosk installations when a long, continuous message must be displayed in a small text box, or where a little on-screen motion may help to attract passing glances. Personally, I like to use scrolling text to create movie-style credits in the "About box" for my applications.

The sample VI, **shown in** ***Figure 3***, demonstrates how scrolling text can be used in an About box VI.

How Does It Work?

In a nutshell, a standard string indicator is moved up the screen, one pixel at a time, simply by decrementing the "top" parameter of the string indicator's

"position" attribute node. When the indicator reaches a predetermined scrolling limit, the "top" value is reset, and the scrolling starts over again. The key to making this work lies in the careful placement and ordering of front panel elements and decorations; as the active string indicator moves up the screen, these elements and decorations mask the

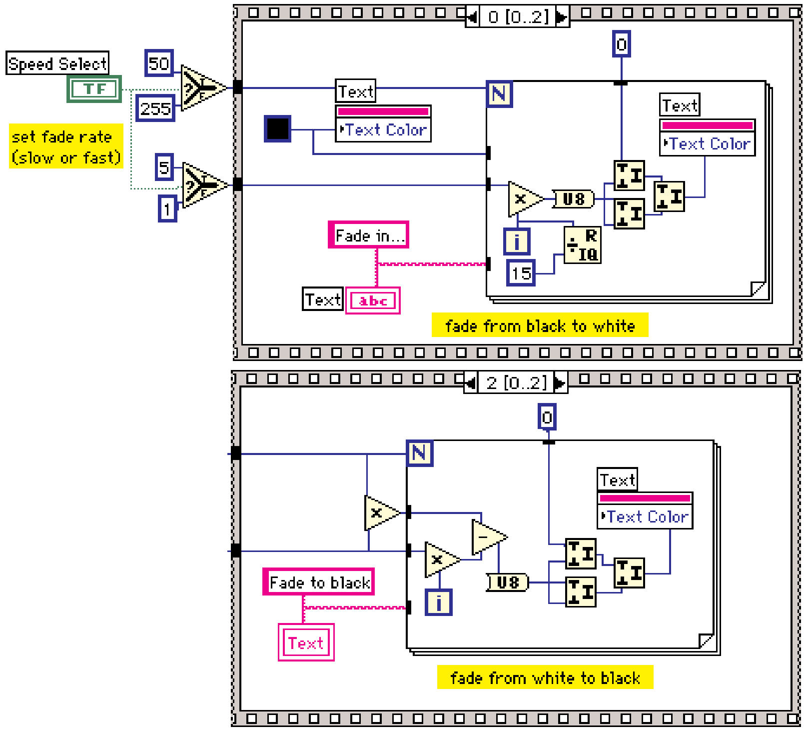


Figure 4: In frame 0 of the sequence structure (upper frame) you can see the logic required to fade text from black to white. After a short time delay in frame 1 (not shown), the sequence proceeds to frame 2 (lower frame) and the text fades back to black. In both cases, RGB "grayscale" values are created by loading R, G, and B values with the same number. A new grayscale value is loaded into the text color attribute for each iteration of the loop.

top part of the message. They also prevent the user from witnessing the abrupt jump that occurs when the scrolling limit is reached and the text is shifted back to the initial position.

SPECIAL EFFECT EXAMPLE 4: Fade to Black Text Effect

[*see* fade2black.vi *in the* Attribute Nodes.llb *file on the LTR resource disk*]

The bonus VI shown in *Figure 4* demonstrates a great trick for creating a nice fading text effect. You may find this effect has a limited number of practical applications in regular LabVIEW GUI design situations, but it makes a nice effect for kiosk installations, tradeshow demos, presentations built using LabVIEW, or even in your application's About box.vi, as an alternative to the scrolling text option outlined previously in Example 3.

How Does It Work?

Using the fact that assigning equal values for R, G, and B to a color attribute will always produce some shade of gray, and the fact that RGB values of [255, 255, 255] yield white, and RGB values of [0,0,0] produce black, we simply cycle through two "For" loops, sequentially setting R,G, and B parameters to each number from 0 to 255, then from 255 to 0. When the 2nd cycle reaches the final destination, the text color is once again black. Superimposed over a black background, the text appears to fade in and then fade out. This effect can also be extended to fade from any color to any other color. I will leave this exercise up to you should the possibilities sound interesting.

A Special Note of Thanks

I would like to take this opportunity to express special thanks to Christophe Salzmann and Danny Lauwers for building the very easy to use platform-specific mouse position subVIs used in two of the demo VIs presented here. It is great to have able programmers posting useful (and free!) LabVIEW extensions to the web for all to use.

A Final Word About Platform Independence

Sharp readers may have noticed that, after citing platform independence as one of the distinct advantages of attribute node usage, I still chose to include platform-specific subVIs in the demo VIs.

Why...?

The platform specific VIs were selected over the picture control mouse attribute for two reasons:

1) They don't require the picture control toolkit, so they are compatible with installations of LabVIEW that don't include the toolkit (before LabVIEW 5.1), and
2) They continue to read the mouse position regardless of the orientation of other components on the front panel. (The picture control mouse attribute will not return a valid reading if an object is placed over the picture control.)

continues on page 14

LabVIEW Special Effects
GUI Techniques Using Attribute Nodes
continued from page 8

In this instance, the platform-specific mouse position VIs were instrumental in demonstrating the specific techniques clearly and directly. But while I defend my choice to use platform-specific VIs in this particular situation, I still encourage all LabVIEW programmers to strive to build platform independent, 100% G applications; they usually need very little (if any) editing to port to another platform and as such are the best insurance in the volatile world of shifting platform popularity. After all, who knows... LabVIEW for Linux might just be your next "big thing"...

Some Final Words

I hope the techniques and demo VIs presented here have provided you with some fresh insights or perhaps inspired your renewed interest into the power and flexibility of the humble attribute node. If you don't generally use attribute nodes very much, I encourage you to tap into some of the hidden potential offered by these highly underrated and easily implemented functions.

About the author:

Dave Ritter is a LabVIEW integrator and interface design specialist for BetterVIEW Consulting who believes that every copy of LabVIEW should come bundled with a copy of Adobe Photoshop!

Dave can be contacted for more LabVIEW interface tips at: solutions@bettervi.com Samples of his unique LabVIEW interface designs can be viewed at: www.bettervi.com/Pages/gallery.htm

LabVIEW™ Technical Resource

THE ONLY LABVIEW SUBSCRIPTION WITH VI SOFTWARE INCLUDED

ORDER FORM

WHAT IS LTR?

LabVIEW Technical Resource (LTR) is a quarterly journal for LabVIEW users and developers available by subscription from LTR Publishing, Inc. Each LTR issue presents powerful LabVIEW tips and techniques and includes a resource disk packed with VI source code, utilities, and documentation. Technical articles on LabVIEW programming methodology, in-depth tutorials, and time-saving tips and techniques address everyday programming issues in LabVIEW.

In its seventh year of publication, LTR has subscribers in over 45 countries and is well-known as a leading independent source of LabVIEW-specific information.

Purchase the LabVIEW Technical Resource CD-ROM Library of Back Issues Version 2.0 and browse this searchable CD-ROM for easy access to over 150 articles and VIs from LTR Vol I-VI.

To subscribe to the LabVIEW Technical Resource or to order the CD-ROM Library of Back Issues, fax this form to LTR Publishing at **(214) 706-0506** or visit the LTR web page at **www.ltrpub.com** to download a free sample issue.

TEL: 214-706-0587 FAX: 214-706-0506

CONTACT INFORMATION

Name ______________________________

Company ______________________________

Address ______________________________

City ____________________ State __________ Zip/Post Code __________

Tel (required) ____________________ FAX ____________________

E-mail ______________________________

ORDER INFORMATION

QTY	MAC/PC	PRODUCT	U.S.	INTL.	EXTENDED PRICE
		1 year subscription (4 issues / 4 diskettes)	$95	$120	
		2 year subscription (8 issues / 8 diskettes)	$175	$215	
		CD-ROM library of back issues (20 issues / over 150 VIs)	$325	$350	
		Server Version CD-ROM library of back issues Ver. 2.0 (5 user license)	$495	$530	
		10 user license Add-On pack (for Server Version)	$295	$325	
		Upgrade CD-ROM to Ver. 2.0 (requires version 1.0)	$89	$99	
		Back issues – [Article Index available at **www.ltrpub.com**]	$25	$30	
				SUBTOTAL	
				TX TAX @ 8.25%	
				TOTAL	

PAYMENT INFORMATION

✔	PAYMENT METHOD
	Check enclosed (U.S. BANK ONLY* – Make check payable to LTR Publishing) (Texas residents please add 8.25% sales tax)
	Bill company / P.O. number required (U.S. Only)
	Visa / MC Card Number Exp.
	Signature
	* Wire information available for international orders

Fill out the form above and Fax it to: 214-706-0506 with your credit card information and signature,
OR fill out the form above and send order form with U.S. check to:

LTR Publishing, Inc., Suite 502, 6060 N. Central Expressway, Dallas, Texas 75206.
Tel: 214.706.0587 • Fax: 214.706.0506 • email: ltr@ltrpub.com

You may also include your own Federal Express or Airborne #. If you are ordering a product for delivery within Texas, please include Texas Sales Tax at 8.25%

WWW.LTRPUB.COM