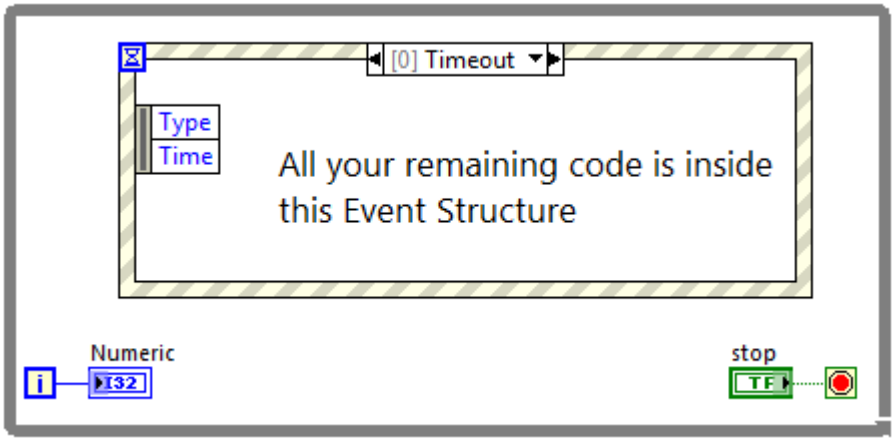
PHYC 500: Introduction to LabView

M.P. Hasselbeck, University of New Mexico

Exercise 3 (v 1.2) Event Structure

A polling structure is a While Loop that executes and updates continuously. If the loop is simply waiting for user input, it performs the same operations over and over. This is a waste of CPU and machine resources. A better approach is an Event Structure.

Open the VI saved in Exercise 2. Go to the Block Diagram, right-click, and select Structures: Event Structure. Use the cursor to surround all the code *inside* the While Loop *except* the stop terminal and the loop count terminal and indicator (see below).

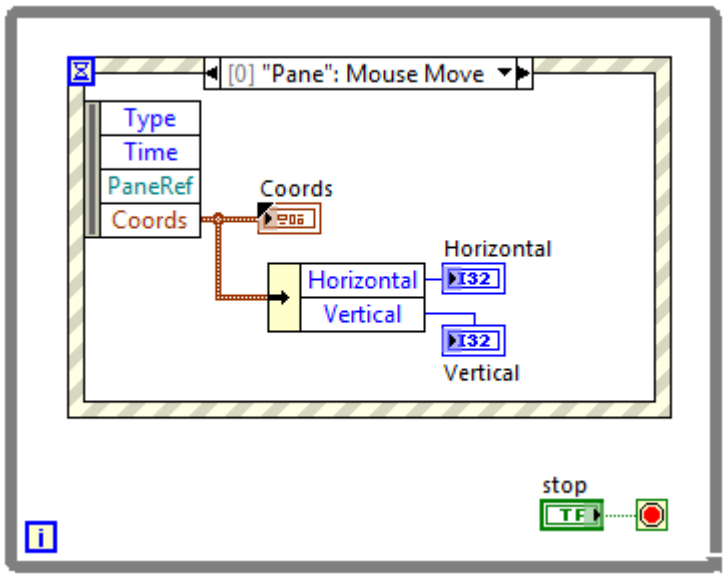


Right-click on the border of the Event Structure and select "Edit Events Handled by this Case". Add three events corresponding to user input for the coefficient *a, b,* and *c* controls on the Front Panel. Click "Add Event", pick the Event Source: Controls: a and then select Events: Value Change. Do the same for controls b and c. Click OK. Run the VI as before with coefficients that produce complex roots. Observe that the loop iteration counter does not increment until one of the coefficients is changed. Also note that a Wait function – as used in the polling architecture – is no longer used or needed.

To further study the mouse position as handled by an Event Structure, open a blank VI and go to the Block Diagram. Right-click and create a While Loop and place a blank Event Structure inside. Create a control button on the stop terminal. Edit the Event Structure by opening "Edit events handled by this case" and selecting Pane: Mouse: Mouse Move. Click OK. The terminals appearing on the left inside border show data that is available in this event window. Drag down the lower edge to reveal the brown **Coords** terminal and create an indicator. This data type is called a cluster, which is a group of one or more different types of data. In this case, it is the Horizontal and Vertical mouse position on the Front Panel. Double-click on the cluster icon and verify that the

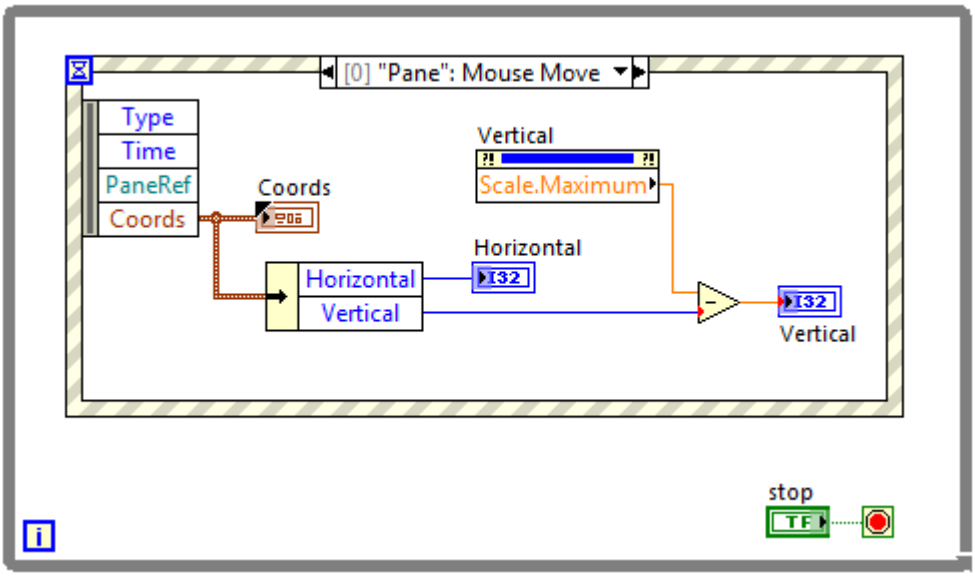
cluster indicator is present on the Front Panel. Run the VI and observe that the mouse position is displayed in real-time.

To independently access data in the cluster, it has to be unbundled. Right-click on the Block Diagram, select Cluster, Class, & Variant: Unbundle By Name. Wire the Coords cluster to the input of this function. Drag it down with the arrow tool to show the two cluster components. Right-click on the output terminals and create two integer output indicators.



Double-click on the Horizontal terminal to locate the indicator on the Front Panel. Right-click the display, Replace: Numeric: Horizontal Fill Slide. Change the vertical indicator to a Vertical Fill Slide. If you run the VI at this point, you will see that the mouse position can greatly exceed the default maxima of both indicators. To fix this, right-click on the indicator and select Properties. Change the scale to a range that is more appropriate, eg. Minimum = 0 and Maximum = 600. Open the Appearance tab and change Height (for Vertical) or Width (for Horizontal) to match the new scale. Run the VI and verify that the two sliders track the mouse movement.

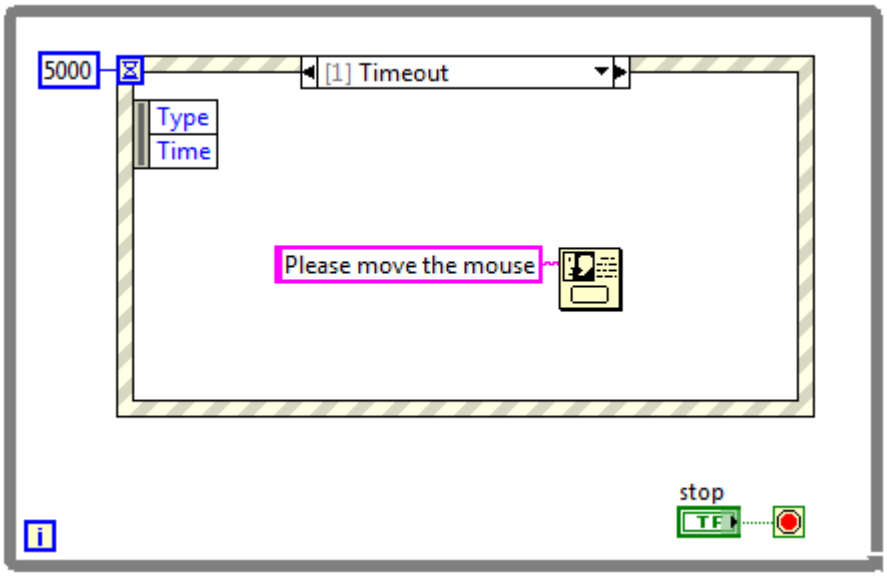
The horizontal slider fills left-to-right following the mouse movement. The vertical slider, in contrast, fills up as the mouse moves down and vice-versa Right-clicking on the vertical slider displays some Fill Options that will not produce a color filling effect that directly tracks the mouse movement. This can be fixed by subtracting the vertical coordinate from the maximum value set on the scale. The VI should be flexible enough to handle this automatically if the scale range is changed. Go to the Block Diagram, right-click on the vertical slider icon, Create Property Node: Scale: Range: Maximum. Place the Property Node function inside the Event Structure. Subtract the Vertical position from the Scale Maximum and wire this to the indicator as shown below:



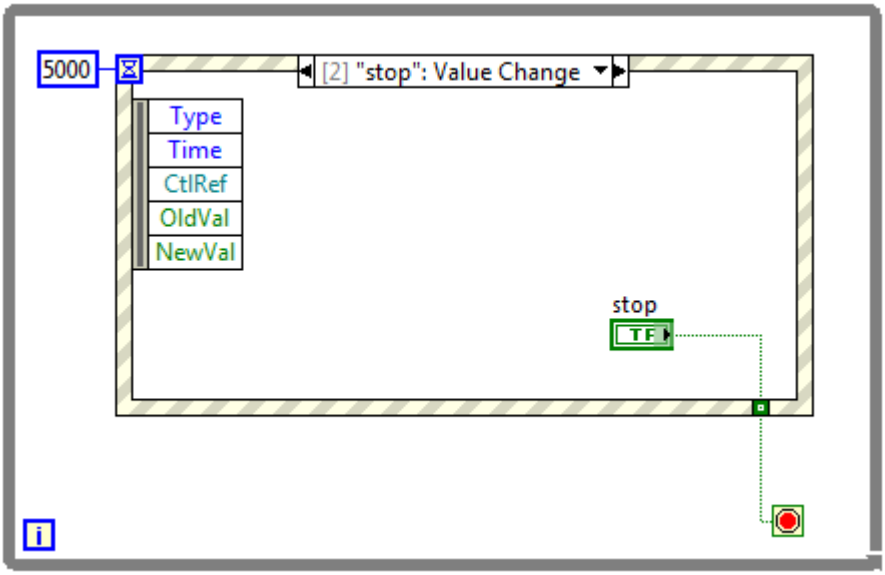
The color in both fill slide indicators should now follow mouse movement when you run the VI. The Property Node can be used to both read and write information to and from VI components. Property Nodes can provide access to many other properties of a selected component.

Note: There are red dots on the input terminals of the subtraction operation and the vertical indicator. These indicate that coercion is taking place. LabView is doing arithmetic with double-precision floating point number and a 32-bit integer. These are different data types. LabView automatically coerces the I32 to a DBL (more memory allocated) and sends the DBL output to an I32 indicator. This results in another coercion. Coercion will not break a VI, but it is usually good programming practice to avoid it. The data can be converted and/or the data representation of an indicator or control can be changed as needed.

Another use for the Event Structure is to set program Timeout. The timeout is infinite by default, but can be adjusted by the programmer. Go to the Block Diagram, right-click on the border of the Event Structure and "Add Event Case". Event Sources: <Application>: Timeout. Click OK. Right-click on the hourglass icon on the upper left of the window and Create Constant. An integer value of –1 corresponds to an infinite timeout; change this to 5000 ms (5 sec). Add a 1-button dialog to alert the user as in the following example:



It is good programming practice to assign an event to the Stop Button of the While Loop. Add an event case to the Event Structure and select the stop control. Place the Stop Button inside this event window as shown:



You should now have three event windows (mouse move, timeout, stop) in the structure. Placing an indicator on the loop iteration counter **[i]** will show that the VI is mostly inactive when it is run.