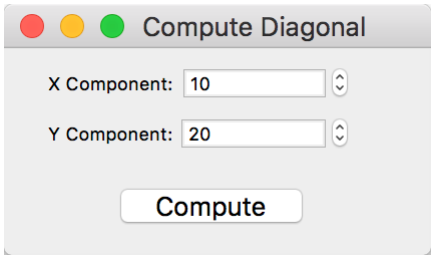
```
   ValDisplay valdisp0,mode=4    // candy stripe
   if( useIgorDraw )
      ValDisplay valdisp0,highColor=(0,65535,0)
   endif
   Button bStop,pos={375,32},size={50,20},title="Abort"
   SetActiveSubwindow _endfloat_
   DoUpdate/W=myProgress/E=1     // mark this as our progress window

   SetWindow myProgress,hook(spinner)=MySpinnHook

   Variable t0= ticks,i
   for(i=0;i<nloops;i+=1)
      PerformLongCalc(1e6)
   endfor
   Variable timeperloop= (ticks-t0)/(60*nloops)

   KillWindow myProgress

   print "time per loop=",timeperloop
End

Function MySpinnHook(s)
   STRUCT WMWinHookStruct &s

   if( s.eventCode == 23 )
      ValDisplay valdisp0,value= _NUM:1,win=$s.winName
      DoUpdate/W=$s.winName
      if( V_Flag == 2 )    // we only have one button and that means abort
         KillWindow $s.winName
         return 1
      endif
   endif
   return 0
End

Function PerformLongCalc(nmax)
   Variable nmax

   Variable i,s
   for(i=0;i<nmax;i+=1)
      s+= sin(i/nmax)
   endfor
End
```

Control Panels and Event-Driven Programming

The CalcDiagDialog function shown under **The Simple Input Dialog** on page IV-144 creates a modal dialog. “Modal” means that the function retains complete control until the user clicks Cancel or Continue. The user can not activate another window or choose a menu item until the dialog is dismissed.

This section shows how to implement the same functionality using a control panel as a modeless dialog. “Modeless” means that the user can activate another window or choose a menu item at any time. The modeless window accepts input whenever the user wants to enter it but does not block the user from accessing other windows.

The control panel looks like this:



The code implementing this control panel is given below. Before we look at the code, here is some explanation of the thinking behind it.

The X Component and Y Component controls are SetVariable controls. We attach each SetVariable control to a global variable so that, if we kill and later recreate the panel, the SetVariable control is restored to its previous state. In other words, we use global variables to remember settings across invocations of the panel. To keep the global variables from cluttering the user's space, we bury them in a data folder located at root:Packages:DiagonalControlPanel.

The DisplayDiagonalControlPanel routine creates the data folder and the global variables if they do not already exist. DisplayDiagonalControlPanel creates the control panel or, if it already exists, just brings it to the front.

We added a menu item to the Macros menu so the user can easily invoke DisplayDiagonalControlPanel.

We built the control panel manually using techniques explained in Chapter III-14, **Controls and Control Panels**. Then we closed it so Igor would create a display recreation macro which we named DiagonalControlPanel. We then manually tweaked the macro to attach the SetVariable controls to the desired globals and to set the panel's behavior when the user clicks the close button by adding the /K=1 flag.

Here are the procedures.

```
// Add a menu item to display the control panel.
Menu "Macros"
   "Display Diagonal Control Panel", DisplayDiagonalControlPanel()
End

// This is the display recreation macro, created by Igor
// and then manually tweaked. The parts that were tweaked
// are shown in bold. NOTE: Some lines are wrapped to fit on the page.
Window DiagonalControlPanel() : Panel
   PauseUpdate; Silent 1   // building window...

   NewPanel/W=(162,95,375,198)/K=1 as "Compute Diagonal"

   SetVariable XSetVar,pos={22,11},size={150,15},title="X Component:"
   SetVariable XSetVar,limits={-Inf,Inf,1},value=
                  root:Packages:DiagonalControlPanel:gXComponent

   SetVariable YSetVar,pos={22,36},size={150,15},title="Y Component:"
   SetVariable YSetVar,limits={-Inf,Inf,1},value=
                  root:Packages:DiagonalControlPanel:gYComponent

   Button ComputeButton,pos={59,69},size={90,20},
                        proc=ComputeDiagonalProc,title="Compute"
EndMacro

// This is the action procedure for the Compute button.
// We created it using the Button dialog.
Function ComputeDiagonalProc(ctrlName) : ButtonControl
   String ctrlName

   DFREF dfr = root:Packages:DiagonalControlPanel
```