Stanford Exploration Project, Report 77, November 17, 1997, pages 431–442

Short Note

Xtpanel update: interactivity from within existing batch programs

Steve Cole & Dave Nichols[1]

keywords: *graphics, algorithm*

INTRODUCTION

Xtpanel (Cole and Nichols, 1992) was developed as a tool for adding a level of interactivity to existing batch seismic processing programs. From an xtpanel-built interface containing buttons, sliders, and other objects, a user can set the values of various processing parameters, then press a button that will launch a conventional background job. The job can be re-run from the panel, to try out different sets of parameters.

While this approach is simpler for the user than having to re-edit parameter files and launch the background jobs manually, it is not particularly efficient. The entire job must be re-run, including the re-reading of input data.

In this paper, we show how to use xtpanel from within a program, to add a greater degree of interactivity. As an example, xtpanel is used to control a data-cube viewing program. Sliders select the planes from the data cube that are displayed on its three faces.

In addition to describing this new use for xtpanel, this paper also summarizes the development of xtpanel since the last report, along with a discussion of what features may be added in the future.

Xtpanel overview

Xtpanel allows one to build a user interface, a panel containing interactive **objects** such as buttons, sliders, text fields, and lists — either directly from the command line, or using a simple scripting language. Each object can have one or more **actions** which take place whenever the object is modified. Actions can execute a system command (a background seismic job, for example), or modify other panel objects in some way.

[1]**email:** not available

Here is a simple example. The panel shown in Figure **??** is generated from the following script:

```
button={ label=QUIT action=QUIT }

button={ label=RUN
   action="Spike n1=128 nsp=1 k1=64 | Bandpass flo=$flo \
           fhi=$fhi phase=$phase | Graph | Tube" }

slider={ name=flo label="Low cut" min=0 max=125 value=10 format="%.2f"
         width=200 }

slider={ name=fhi label="High cut" min=0 max=125 value=30 format="%.2f"
         width=200 }

choice={ name=phase label=Phase value=0 orientation=horizontal
   item={ label=Zero value=0 }
   item={ label=Minimum value=1 }
}
```

In just a few lines, we have built an interactive frontend to an existing seismic filtering program. The panel contains several different xtpanel objects — buttons, sliders, and a `choice` object. Most of the objects in this example have no action; it is the `RUN` button that puts the values of the objects together (referring to each object by $name) and then calls the bandpass filter program. Actually it calls several programs — `Spike` creates a filter with a single impulse; `Bandpass` does the filtering; `Graph` makes a graph of the function, and `Tube` displays the graph on the workstation screen.

While building frontends in this way can be useful, often the overhead of having to re-run the entire job makes this level of interactivity unsatisfying. In the following section, we describe a way to use the interactivity of xtpanel from within programs.

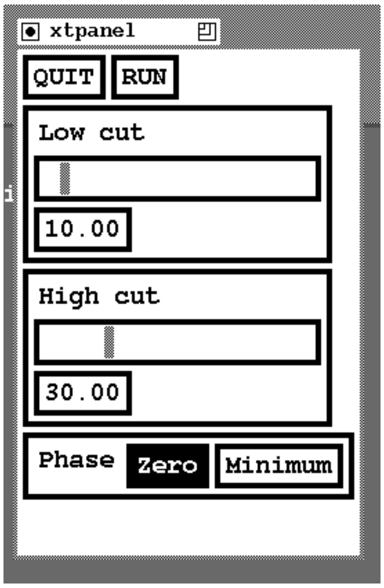

Figure 1: An xtpanel interactive frontend to a bandpass filter program. If you are reading this report on CD-ROM, you can click on the button at the end of this caption to run this example. steve2-filt [NR]

Interactivity with xtpanel

Using xtpanel from within a seismic processing program offers some advantages. In the filtering example above, because the entire process is re-run each time a parameter is changed, a lot of extra work is done. Each time the `RUN` button is pressed, the input dataset (containing a single impulse) is recreated. Then the filter program must read in the data, as well as the parameters necessary to run the job. In such a trivial example, this extra work does not take much time. But in larger tasks, the extra overhead required to completely re-run a job may make the interactive performance poor.

To solve this problem, we devised a way to use xtpanel from within a processing program. After reading in its data and getting set up to run, a program brings up an xtpanel containing objects that specify some of the program parameters. Then the program waits for input from the xtpanel. When input is received, the program executes, and then waits for additional panel input. Additional panel input causes the program to be re-run, but without having to do all the overhead that was done at the start of the program.

The first program to use this interaction is a seismic data cube viewing program called `Cubeplot`. Cubeplot displays a perspective view of a seismic data cube. Displayed on the three faces of the cube are slices taken from within the data volume. In its standard batch mode of execution, the user specifies on the command line the three slices to be shown. If, on the command line, the user specifies `popup=y`, then a panel containing three sliders appears. As the user changes a slider, a different data slice is displayed on one of the three cube faces.

The xtpanel script is built into the C language source code for Cubeplot. Here is the script:

```
button={ label=Quit action="PRINTQUIT" action=QUIT }
vbox={
  hbox={ name=noborder width=400
    message={ value=axis-1 }
    scrollbar={ label="axis 1" name=pan1 min=0 max=N1 format="%.0f"
      width=300 value=FRAME1
      action="ASSIGN val1 $val"
      action="PRINT frame1=$(pan1) frame2=$(pan2) frame3=$(pan3)\n"
    }
    message={ name=val1 value=FRAME1  }
  }
  hbox={ name=noborder width=400
    message={ value=axis-2 }
    scrollbar={ label="axis 2" name=pan2 min=0 max=N2 format="%.0f"
      width=300 value=FRAME2
      action="ASSIGN val2 $val"
      action="PRINT frame1=$(pan1) frame2=$(pan2) frame3=$(pan3)\n"
    }
    message={ name=val2 value=FRAME2  }
  }
```

```
  hbox={ name=noborder width=400
    message={ value=axis-3 }
    scrollbar={ label="axis 3" name=pan3 min=0 max=N3 format="%.0f"
      width=300 value=FRAME3
      action="ASSIGN val3 $val"
      action="PRINT frame1=$(pan1) frame2=$(pan2) frame3=$(pan3)\n"
    }
    message={ name=val3 value=FRAME3  }
  }
}
```

However, the user can substitute a different script, containing whatever program parameters are of interest. Here are the instructions that appear in the on-line program documentation:

```
popup   Specifying popup=y brings up an xtpanel (if you have xtpanel
        installed) with three sliders. Moving these sliders changes
        the frames plotted on the three cube faces. If you pipe
        the output of Cubeplot to "Xtpen cachepipe=n" you will see
        the display update as the sliders are moved.
        You can specify your own xtpanel script file by doing
        popup_file=filename.
```

The program then calls two subroutines: `popup_start` brings up the xtpanel. This is called after the program's preliminary work (reading in the input data, getting parameter values, etc.) has been done. A second routine, `popup_check` is called after each pass through processing the data. This routine waits for output from the xtpanel. It then adds the new output to the table of parsed command line arguments. I.e. if the user moves a slider and the xtpanel prints "frame1=100" then after `popup_check` it is as though "frame1=100' had been specified on the command line. When this routine returns the program reprocesses the data (in the case of Cubeplot case plotting a new figure) using the new parameters.

Users interested in the details of these routines are referred to the CD-ROM version of this report, where the source code for Cubeplot and the popup routines is contained. The most important point to make about the source is that the modifications to the Cubeplot program were quite small — just two subroutine calls, and a few lines to interpret the new parameter values that come back from the panel. Interaction can easily be added to any program in this way.

Figure **??** displays a typical interactive session using the built-in xtpanel interaction of `Cubeplot`. If you are reading the report on CD-ROM, pressing the button at the end of the figure caption will bring up this example. Moving the sliders will change the display.

It is worth noting that although Cubeplot is a C language program, the xtpanel popup facility can just as easily be used from within Fortran and Ratfor programs.

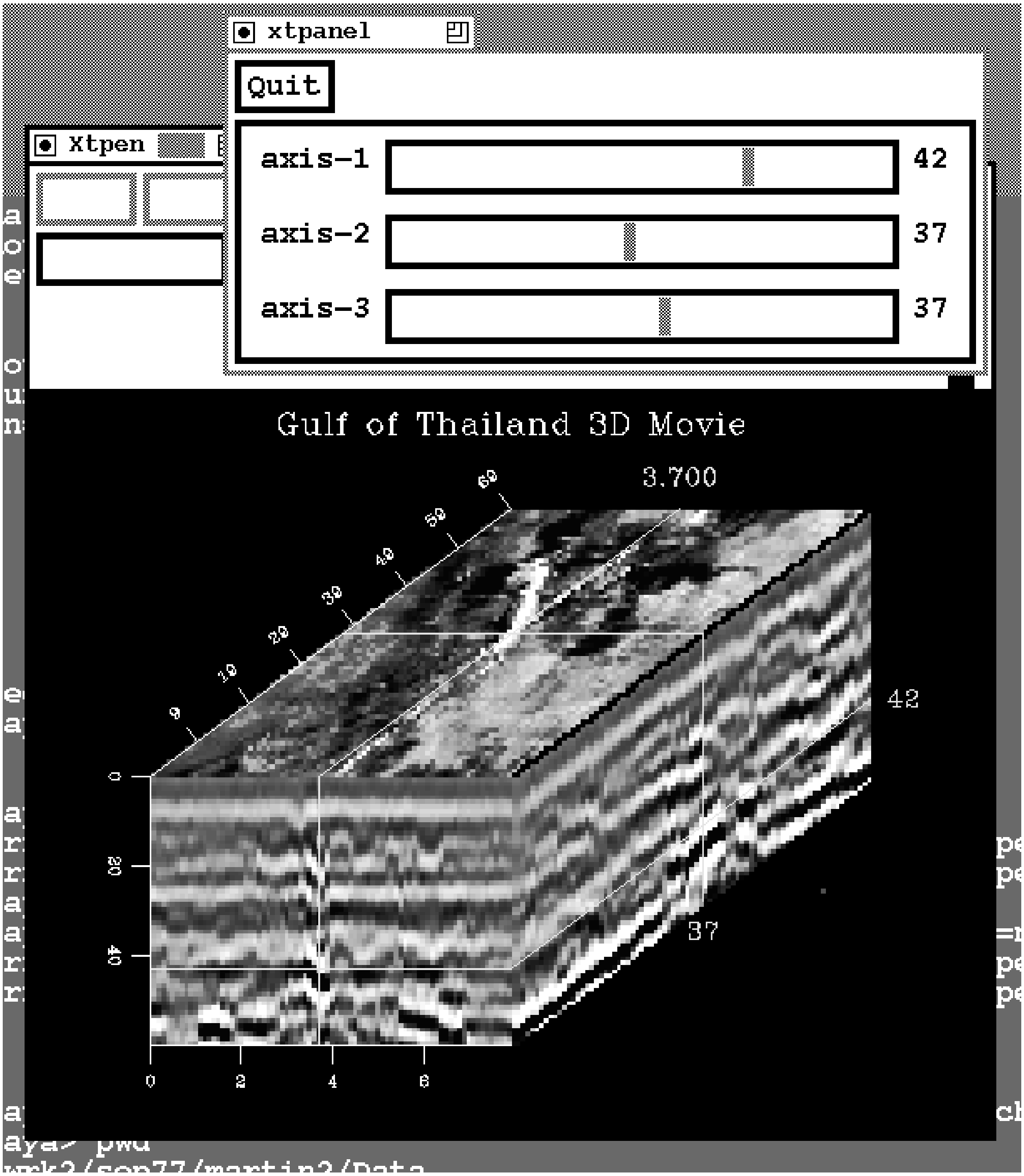

Figure 2: Interactive Cubeplot session using xtpanel. The xtpanel is brought up from within the program. Each time a slider is moved, a subroutine is run to re-draw the image. CD-ROM readers can click on the button at the end of this caption to run Cubeplot. steve2-cubeplot [NR]

Xtpanel update and future possibilities

We wrote xtpanel to fill a need, to add interactivity to the large library of non-interactive seismic processing programs available at SEP. In addition to making day-to-day data processing chores easier, this interactivity is also useful in building tutorials and interactive documents, such as this report. While other public-domain interface-building software exists, we believe that no other simple package is flexible enough to meet these needs.

Because it does not depend on other SEP software or geophysical software in general, xtpanel has been distributed through several public-domain source channels. Feedback from interested users has helped xtpanel to evolve to a reliable state in a short period of time.

There are several minor ways in which xtpanel could be improved in the future. Support of Motif in addition to the MIT Athena widget set would make xtpanel compatible with the software environments of more potential users. Because of the modular way in which xtpanel has been written, and because the Athena widgets and Motif have many features in common, adding Motif support would not be very difficult.

Another promising possibility is to be able to put images (pixmaps) in the background of xtpanel objects. If this were possible, one could, for example, very quickly build seismic tools that incorporate data picking, by having an image of the data in the background of certain interactive objects.

REFERENCES

Cole, S., and Nichols, D., 1992, Xtpanel: An interactive panel builder: SEP–**75**, 497–520.

```
I grabbed a copy of your xtpanel, compiled it and tried it out. It
does exactly what I've been looking for. I was able to begin building
screens immediately.
I've looked briefly at a couple of commercial products and I
have to admit they are pretty slick, but the simplicity and
intuitiveness of xtpanel makes it highly desireable to me (plus the
price is right:-)

=====================================================

I just got your latest release of xtpanel and have been having a
lot of fun with it.  Being a sys admin type, I know a lot about
shell programming, but haven't had time to learn how to program
with X, so xtpanel is something I intend to use a lot.

=====================================================

Dear Steve and Dave,

I just grabbed the most recent copy of your xtpanel directory
(from sepcd4) and compiled it.  It's great.  Everything
works so far.  The documentation makes sense.  Thanks for making
it detachable from the rest of seplib.  It'll get more use
that way.

=====================================================

Steve and Dave,

I must congratulate you again on xtpanel.  It definitely fills a need.
I wrote a single shell file that serves as a front-end for half a dozen
programs with getch/fetch type I/O.  I rewrote my own version of your
lister program just for fun.  Another fellow here is going to write a
front-end for "gslib," a geostatistics package written at Stanford, in
the Pet. Eng. Dept., I think.  You might want to show xtpanel to the
authors.  They use ASCII parameter files, with more-or-less free format.
Shouldn't take more than an hour to come up with something pretty good.

=====================================================

The product is great for a non-C programmer like me. And it makes it very
easy to develop a friendly interface destined to a UNIX novice

=====================================================

I fired this up and was pleasantly surprised.  It looks like it's
possible to do some reasonable things with it (and, unlike some of the
other packages available, it's small and easy to use).

=====================================================

Hello Steve and/or Dave,

        Recently I fetched 'xtpanel' from the net and I first have to
```

```
        congratulate you with the product! The program is very useful
        for prototyping (simple) user interfaces and for making fancy
        shell interfaces. I linked it with the Xaw3d library (as someone
        suggested on the net) and it looks great!

=====================================================

Hello Steve,

Let me first express my appreciation for such an incredible tool.  I
am a computer programmer for the US Air Force and I have several
commercial (expensive) GUI builders at my disposal.  I picked up
xtpanel while browsing around the internet and I have been building my
user interfaces with it ever since.  It is more complete and extensive
than any of the commercial packages that I have been using.
```