Subject: Re: The proper way of catching mouse button events from a draw widget?
Posted by David Fanning on Wed, 18 Oct 2006 12:49:59 GMT
View Forum Message <> Reply to Message

Braedley writes:

> Reading through the widget_draw documentation, they suggest that cursor
> not be used for this type of thing, but instead use mouse events. This
> is despite the fact that using cursor hasn't caused any problems at all
> in my widget programs.

Well, any problems you have noticed yet. :-)

Typically the reported cursor location is 2-3 pixels off from
where you *think* you clicked. Not a big deal most of the time,
but a HUGE annoyance when you are trying to do things like
select a resizing square or something else that requires
precision pointing.

> However, I would still like to use the proper
> implementation whenever possible. So the question becomes, how do I
> grab mouse events only when I want them, and then return to the proper
> point in the current event handler?

Uh, I'm not sure you fully understand how widget event
handlers work. There is no "returning to the proper
point" in an event handler. Events are one-shot deals.
An event happens, it is handled. If another event occurs
while the first event is being handled, the event gets
queued up so it can be handled when the first event
handler is finished, etc. Events are processed one after
the other, consecutively.

To see how widget events might be used to select the
boundary of a plot, have a look at ZPLOT:

http://www.dfanning.com/programs/zplot.pro

Here the boundaries are chosen by the user clicking
and dragging on the plot. To restore the plot to its
original boundaries, just click and release somewhere
inside the plot.

Cheers,

David
--
David Fanning, Ph.D.

Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: The proper way of catching mouse button events from a draw widget?
Posted by Allan Whiteford on Wed, 18 Oct 2006 13:26:06 GMT
View Forum Message <> Reply to Message

Braedley,

I don't think the event driven model is setup to allow the same type of programming. I'd guess you'd need to maintain status elsewhere that you're expecting the next two events to be button clicks on the graph. You could have your draw widget ignore any button clicks made when this switch isn't set. This probably replicates what you're doing at the moment but is much less convenient to code than just an inline call to cursor when you know someone is about to click.

I don't think you can return to the proper point in the event handler. So half your code will need to be written before you're expecting the button clicks, then get the clicks and store them somewhere and finally write the last half of your code after you have both clicks. Something like:

```
pro allan_e,event
 widget_control,event.handler,get_uvalue=info

        if event.id eq info.button then begin
         info.x1=!values.f_nan
         info.x2=!values.f_nan
                info.selecting=1
  widget_control,event.handler,set_uvalue=info
                return
        endif

        if event.id eq info.plotwindow $
           and event.press eq 1 $
           and info.selecting eq 1 then begin
         widget_control,info.plotwindow,get_value=wid
                wset,wid
         !x=info.x_store
                !y=info.y_store
                xval=(convert_coord(event.x,event.y,/device,/to_data))[0]
                plots,[xval,xval],!y.crange

                if not finite(info.x1) then  $
                 info.x1=xval   $
  else if not finite(info.x2) then $
```

```
                info.x2=xval

          widget_control,event.handler,set_uvalue=info
         endif

 if finite(info.x1) and finite(info.x2) then begin
          print,info.x1,info.x2
                info.selecting=0
          info.x1=!values.f_nan
          info.x2=!values.f_nan
  widget_control,event.handler,set_uvalue=info
         endif

end

pro allan_w

         xvals=findgen(100)
         yvals=sqrt(xvals)

 tlb=widget_base(title='Example',/column)
          plotwindow=widget_draw(tlb,xsize=640,ysize=480,/button_event s)
 button=widget_button(tlb,value='Push to select a range')
         widget_control,tlb,/realize

         widget_control,plotwindow,get_value=wid
         wset,wid
         plot,xvals,yvals
         x_store=!x
         y_store=!y

         info={ plotwindow:plotwindow, $
          button:button,  $
                xvals:xvals,  $
                yvals:yvals,  $
                x1:!values.f_nan, $
                x2:!values.f_nan, $
                selecting:0,  $
                x_store:x_store, $
          y_store:y_store  }
 widget_control,tlb,set_uvalue=info
         xmanager,'allan_w',tlb,event_handler='allan_e'
end
```

Thanks,

Allan

Braedley wrote:
> In many of my programs (be it command line or widgets), I often need
> the user to enter a bounds of a particular graph by clicking on it.
> With basic windows, it's just using cursor like so:
>
> cursor, x1, junk, /data
> plots, [x1, x1], !y.crange, /data, color=254 ;color=red
> cursor, x2, junk, /data
> plots, [x2, x2], !y.crange, /data, color=254 ;color=red
>
> Reading through the widget_draw documentation, they suggest that cursor
> not be used for this type of thing, but instead use mouse events. This
> is despite the fact that using cursor hasn't caused any problems at all
> in my widget programs. However, I would still like to use the proper
> implementation whenever possible. So the question becomes, how do I
> grab mouse events only when I want them, and then return to the proper
> point in the current event handler?
>
> Braedley
>

Subject: Re: The proper way of catching mouse button events from a draw widget?
Posted by Braedley on Wed, 18 Oct 2006 14:01:28 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Braedley writes:
>
>> Reading through the widget_draw documentation, they suggest that cursor
>> not be used for this type of thing, but instead use mouse events. This
>> is despite the fact that using cursor hasn't caused any problems at all
>> in my widget programs.
>
> Well, any problems you have noticed yet. :-)
>
> Typically the reported cursor location is 2-3 pixels off from
> where you *think* you clicked. Not a big deal most of the time,
> but a HUGE annoyance when you are trying to do things like
> select a resizing square or something else that requires
> precision pointing.
>
>> However, I would still like to use the proper
>> implementation whenever possible. So the question becomes, how do I
>> grab mouse events only when I want them, and then return to the proper
>> point in the current event handler?
>
> Uh, I'm not sure you fully understand how widget event

> handlers work. There is no "returning to the proper
> point" in an event handler. Events are one-shot deals.
> An event happens, it is handled. If another event occurs
> while the first event is being handled, the event gets
> queued up so it can be handled when the first event
> handler is finished, etc. Events are processed one after
> the other, consecutively.
>
> To see how widget events might be used to select the
> boundary of a plot, have a look at ZPLOT:
>
>   http://www.dfanning.com/programs/zplot.pro
>
> Here the boundaries are chosen by the user clicking
> and dragging on the plot. To restore the plot to its
> original boundaries, just click and release somewhere
> inside the plot.
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

I'm well aware of how event driven programming works, so I probably should have been a little more precise. What I should have asked is how would I go about creating a function or procedure that would perform the same action as a call to cursor, but follow the guidelines for draw widgets.

In any case, the example you provided, as well the example provided by Allan are good starting points. It's a bit unfortunate that the example I provided is just that: an example. It isn't indicative of the scope of the entire program in question. Due to the flexible nature of the program, a user may include procedures that need cursor like calls that would exceed the scope of the basic program.

Thanks anyways

Braedley

Subject: Re: The proper way of catching mouse button events from a draw widget?
Posted by David Fanning on Wed, 18 Oct 2006 14:25:23 GMT

View Forum Message <> Reply to Message

Braedley writes:

> I'm well aware of how event driven programming works, so I probably
> should have been a little more precise. What I should have asked is
> how would I go about creating a function or procedure that would
> perform the same action as a call to cursor, but follow the guidelines
> for draw widgets.

Well, presumably you heard that it can't be done. :-)

> In any case, the example you provided, as well the example provided by
> Allan are good starting points. It's a bit unfortunate that the
> example I provided is just that: an example. It isn't indicative of
> the scope of the entire program in question. Due to the flexible
> nature of the program, a user may include procedures that need cursor
> like calls that would exceed the scope of the basic program.

It wasn't my intention to make you defensive, but
I think your approach to the problem is going to make
it very difficult to write a "proper" widget program,
whatever that is. Of course, all kinds of programs can
be (and are) written in IDL. Some just work better
than others.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: The proper way of catching mouse button events from a draw widget?
Posted by Allan Whiteford on Wed, 18 Oct 2006 15:54:41 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Braedley writes:
>
>
>> I'm well aware of how event driven programming works, so I probably
>> should have been a little more precise. What I should have asked is
>> how would I go about creating a function or procedure that would

>> perform the same action as a call to cursor, but follow the guidelines
>> for draw widgets.
>
>
> Well, presumably you heard that it can't be done. :-)
>
>

I think it perhaps could be... almost:

If we get the user subroutine to supply its own function name and a magical index to the draw widget event handler then store all it's variables before returning.

After the draw widget has collected the min and max x-range it can use a call_function to call back the original function which will take the index and essentially implement an entry point via a goto and then restore all the saved variables. It can then carry on with the min and max x-range.

I think that will pretty much give the same functionality as using an inline call to cursor. You still need to protect against someone not clicking on the draw widget twice but presumably the original cursor calls were open to that as well. I've not used cursor in this millennium so I don't know what happens if you click on a button when cursor is expecting some coordinates.

However, my 'solution' is repulsive and there is no way I'm supplying example code for the above suggested catastrophe. One day far from now in a job interview someone might put it down in front of me and ask if I actually wrote it.

I'm also of the opinion that if the above is the way to do it then it's almost indistinguishable from David's assertion that "it can't be done". However if you're in the situation where you have code you can't refactor for whatever reason and using cursor is giving you the wrong answer then maybe it's easier (but certainly not better) than re-writing everything to function within how IDL is designed to do such things.

Thanks,

Allan

Subject: Re: The proper way of catching mouse button events from a draw widget?
Posted by David Fanning on Wed, 18 Oct 2006 16:04:38 GMT
View Forum Message <> Reply to Message

Allan Whiteford writes:

>> Well, presumably you heard that it can't be done. :-)
>>
>>
>
> I think it perhaps could be... almost:
>
> If we get the user subroutine to supply its own function name and a
> magical index to the draw widget event handler then store all it's
> variables before returning.
>
> After the draw widget has collected the min and max x-range it can use a
> call_function to call back the original function which will take the
> index and essentially implement an entry point via a goto and then
> restore all the saved variables. It can then carry on with the min and
> max x-range.

A GOTO!? Your honor, I rest my case.

> However, my 'solution' is repulsive and there is no way I'm supplying
> example code for the above suggested catastrophe. One day far from now
> in a job interview someone might put it down in front of me and ask if I
> actually wrote it.

I'm certain we agree on this. -)

> I'm also of the opinion that if the above is the way to do it then it's
> almost indistinguishable from David's assertion that "it can't be done".

You are just trying to get on my good side, here.

> However if you're in the situation where you have code you can't
> refactor for whatever reason and using cursor is giving you the wrong
> answer then maybe it's easier (but certainly not better) than re-writing
> everything to function within how IDL is designed to do such things.

I'm not so sure, and anyway, he claimed that he wanted
to learn the "proper" way of doing this. This abomination
is certainly not that, IMHO. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Sepore ma de ni thui. ("Perhaps thou speakest truth.")