Subject: Re: border around draw widget
Posted by Rick Towler on Thu, 28 Sep 2006 17:17:24 GMT
View Forum Message <> Reply to Message

I think we have strayed way off on this one... While JD's suggestion is clever, from a usability perspective I don't think it is as effective as a colored border or a color shift of the image. And adding a border is trivial.

If you haven't done this already, you'll want all of your draw widgets to share the same click event handler. You mention in your original post that your displaying gamma-scans. I'll assume these are images of some sort. I'll further assume you are using direct graphics and are displaying the image using TV (I know in reality you are using one of David's or Liam's improved versions).

The one thing I don't know is how you are storing your application data. You are going to need to keep the selection state of each draw widget and a copy of the image displayed in the widget. In my example I put them in a structure with the fields "image" and "selected" and store that in each draw widgets UVALUE. You may already have this data stored someplace else.

```
pro drawClick_event, ev

    WIDGET_CONTROL, ev.id, GET_UVALUE=thisData, /NO_COPY
    WIDGET_CONTROL, ev.id, GET_VALUE=thisWindow

    if (thisData.selected) then begin
       ;  window is currently selected, deselect
       WSET, thisWindow
       TV, thisData.image
       thisData.selected = 0
    endif else begin
       ;  window is currently not selected
       WSET, thisWindow
       OPLOT, [0,0,1,1,0], [0,1,1,0,0], COLOR=255, THICK=4
       thisData.selected = 1
    endelse

    WIDGET_CONTROL, ev.id, SET_UVALUE=thisData

end
```

You'll notice that the border isn't perfect but it is close. Also, you'll want to modify the COLOR value accordingly.

-Rick

Laurens wrote:
> JD Smith wrote:
>> On Wed, 27 Sep 2006 23:35:34 +0200, Laurens wrote:
>>
>>> Thanks very much for that explanation!
>>> Could you tell me how to make such a widget-object? It sounds like
>>> something I was already thinking about...
>>
>> It sounds fancier than it is. It's basically an object, which:
>>
>> 1. Sets up a widget heirarchy in the normal way (usually in its Init
>> method).
>> 2. Saves its "state" information not in a structure in a UVALUE but in
>> the class data itself (e.g. self.*).
>> 3. Calls XManager (often, but not necessarily, in its Init method) to
>> generate events on that widget heirarchy.
>> 4. Uses the trick I outline to inject the events flowing forth from
>> the widgets created to some class method (often named "Event").
>>
>> The main advantages of this method:
>>
>> 1. You get state information "for free", quite nicely mapped to class
>> data.
>> 2. You automatically avoid common blocks for state info, with their
>> associated collision risks if multiple identical widgets run at the
>> same time.
>> 3. You are never left with state information "in the air", if you use
>> /NO_COPY to be efficient when retrieving your state structure from
>> a UVALUE. This greatly aids debugging, since crashes to the code
>> usually can be recovered from with a simple RETALL.
>> 4. You quickly realize that the normal event flow embodied in "normal"
>> widget prgramming is limiting, and can roll your own communication
>> among objects that suits your needs. This is particularly useful
>> if you have many different perhaps unrelated application components
>> that need to communicate with eachother.
>>
>> A schematic usage would be:
>>
>> oDraw=obj_new('SelectableDrawPane',base)
>>
>> which would place a new compound widget into base. It might implement
>> some methods "Select" and "DeSelect", or you could have it trap the
>> selection "clicks" and automagically select/deselect itself. Once you
>> have the apparatus in place, you can then have fun implementing other

>> methods for your object, drawing and erasing, etc.
>>
>> I should note that none of this is necessary to use the "base on top
>> of a base" trickery I outlined before, it just makes it easier and more
>> powerful.
>>
>> JD
>>
> err...well, I'll try hehe; If I understand it correctly, this implies
> writing code in the GUI.pro file, with as disadvantage that I can't use
> my .prc file to regenerate GUI?
> That's some strange behaviour I noticed earlier...if you change position
> of a widget and save the GUI, all self-written code is simply gone :S
>
> If I've created that object, where could I change its structure, like
> the select and deselect functions?
> Sorry for the quite explicit way of asking, but hey I'm not as
> experienced in writing IDL as you guys huh (will one ever be haha), so
> I'm just trying to learn from it...
>
> Thanks though for what you've brought up on ideas so far...
>
> Laurens
>
> Cheers David :) It's quite funny, I live in the Netherlands and when we
> use the word "cheers", its when we take a beer haha; so every "cheers"
> underneath your msgs lets me think you're having quite a good time lol.
