Subject: Re: border around draw widget
Posted by Michael Galloy on Wed, 27 Sep 2006 14:10:33 GMT
View Forum Message <> Reply to Message

Laurens wrote:
> I have a couple of draw widgets displaying various gamma-scans. Now the
> user has to select two of them by clicking on them; done.
> But now I want to let the user know where he clicked, so is it possible
> to draw some sort of red border of a few pix around the clicked
> draw-widget? I know they have a frame property, but that's just too thin
> and doesn't attract attention...

What about some polylines on the outside edge of the draw widget? Direct
graphics or object graphics?

Mike
--
www.michaelgalloy.com

Subject: Re: border around draw widget
Posted by Laurens on Wed, 27 Sep 2006 14:28:38 GMT
View Forum Message <> Reply to Message

Michael Galloy wrote:
> Laurens wrote:
>> I have a couple of draw widgets displaying various gamma-scans. Now
>> the user has to select two of them by clicking on them; done.
>> But now I want to let the user know where he clicked, so is it
>> possible to draw some sort of red border of a few pix around the
>> clicked draw-widget? I know they have a frame property, but that's
>> just too thin and doesn't attract attention...
>
> What about some polylines on the outside edge of the draw widget? Direct
> graphics or object graphics?
>
> Mike
> --
> www.michaelgalloy.com

Yeah but with 14 draw widgets on screen, isn't that a bit of a nasty
solution?

Subject: Re: border around draw widget
Posted by Michael Galloy on Wed, 27 Sep 2006 14:52:34 GMT
View Forum Message <> Reply to Message

Laurens wrote:
> Michael Galloy wrote:
>> Laurens wrote:
>>> I have a couple of draw widgets displaying various gamma-scans. Now
>>> the user has to select two of them by clicking on them; done.
>>> But now I want to let the user know where he clicked, so is it
>>> possible to draw some sort of red border of a few pix around the
>>> clicked draw-widget? I know they have a frame property, but that's
>>> just too thin and doesn't attract attention...
>>
>> What about some polylines on the outside edge of the draw widget?
>> Direct graphics or object graphics?
>>
>> Mike
>> --
>> www.michaelgalloy.com
>
> Yeah but with 14 draw widgets on screen, isn't that a bit of a nasty
> solution?

It depends. If you make an object representing a draw widget, then it's just a matter of calling odraw->select and odraw->unselect (once you figure out what goes in those methods).

Mike
--
www.michaelgalloy.com

Subject: Re: border around draw widget
Posted by Laurens on Wed, 27 Sep 2006 15:02:52 GMT
View Forum Message <> Reply to Message

Michael Galloy wrote:
> Laurens wrote:
>> Michael Galloy wrote:
>>> Laurens wrote:
>>>> I have a couple of draw widgets displaying various gamma-scans. Now
>>>> the user has to select two of them by clicking on them; done.
>>>> But now I want to let the user know where he clicked, so is it
>>>> possible to draw some sort of red border of a few pix around the
>>>> clicked draw-widget? I know they have a frame property, but that's
>>>> just too thin and doesn't attract attention...
>>>
>>> What about some polylines on the outside edge of the draw widget?
>>> Direct graphics or object graphics?
>>>
>>> Mike

>>> --
>>> www.michaelgalloy.com
>>
>> Yeah but with 14 draw widgets on screen, isn't that a bit of a nasty
>> solution?
>
> It depends. If you make an object representing a draw widget, then it's
> just a matter of calling odraw->select and odraw->unselect (once you
> figure out what goes in those methods).
>
> Mike
> --
> www.michaelgalloy.com

I make my draw widgets simply by drawing them in the gui and link an
event to it; should that be done in another way?

Subject: Re: border around draw widget
Posted by David Fanning on Wed, 27 Sep 2006 16:47:01 GMT
View Forum Message <> Reply to Message

Laurens writes:

> I make my draw widgets simply by drawing them in the gui and link an
> event to it; should that be done in another way?

OF COURSE it should be done another way. The difficulty
is in getting ITTVIS to realize it. :-)

Cheers,

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

Subject: Re: border around draw widget
Posted by Laurens on Wed, 27 Sep 2006 17:31:29 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Laurens writes:
>
>> I make my draw widgets simply by drawing them in the gui and link an

>> event to it; should that be done in another way?
>
> OF COURSE it should be done another way. The difficulty
> is in getting ITTVIS to realize it. :-)
>
> Cheers,
>
> David

lol.... is it an option to make one rectangular widget and, depending on the selected draw-widget, change position of that rectangular everytime they click on a draw-widget? Or is that even nastier? hehe...

Subject: Re: border around draw widget
Posted by JD Smith on Wed, 27 Sep 2006 18:06:38 GMT
View Forum Message <> Reply to Message

On Wed, 27 Sep 2006 15:42:43 +0200, Laurens wrote:

> Hi,
>
> I have a couple of draw widgets displaying various gamma-scans. Now the
> user has to select two of them by clicking on them; done.
> But now I want to let the user know where he clicked, so is it possible
> to draw some sort of red border of a few pix around the clicked
> draw-widget? I know they have a frame property, but that's just too thin
> and doesn't attract attention...

You could make the frame thicker. Something like:

```
IDL> b=widget_base(/ROW)
IDL> t=widget_draw(b,xsize=200,ysize=200,FRAME=10)
IDL>  b2=widget_base(b,xsize=200,ysize=200,xpad=10,ypad=10,/COLUMN )
IDL> t2=widget_draw(b2,xsize=200,ysize=200)
IDL> widget_control, b,/realize
```

Unfortunately with this setup, you'll have to kill and re-create everything whenever the user selects a new draw widget (since FRAME's are only possible when the widget is created). Not ideal. You'll probably want to use UPDATE=0/1 to avoid flickering.

Here's a trick that avoids all that killing/recreating. Use a bulletin board base (no /ROW or /COLUMN), and then layer two bases inside it: one which holds your draw widget, offset by 10 pixels in x an y, and one empty base which has FRAME=10 set, whose only purpose in life is to display that frame. Simply map and unmap the empty draw base to add/remove the frame as necessary. Something like:

```
IDL> b=widget_base()
IDL> b1=widget_base(b,xsize=200,ysize=200,xpad=10,ypad=10)
IDL> t=widget_draw(b1,xsize=200,ysize=200)
IDL> b2=widget_base(b,xsize=200,ysize=200,FRAME=10)
IDL> widget_control, b,/realize
IDL> widget_control, b2,map=0 ; remove frame
IDL> widget_control, b2,map=1 & widget_control, b1,map=1 ; add frame back
```

To make it easier, wrap this functionality up in a compound "frame-toggle-draw" widget of some sort, and then layout as many of these as you need. I might instead make it an object widget for additional convenience (so I can pass it an image to draw, tell it to erase, etc.), but a regular compound widget would work as well. Then something like:

```
widget_control,frame_toggle_draw_widget_id,SET_VALUE=0 ; turn frame off
```

could be enough to "de-select" that draw.

JD

Subject: Re: border around draw widget
Posted by Laurens on Wed, 27 Sep 2006 21:35:34 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> On Wed, 27 Sep 2006 15:42:43 +0200, Laurens wrote:
>
>> Hi,
>>
>> I have a couple of draw widgets displaying various gamma-scans. Now the
>> user has to select two of them by clicking on them; done.
>> But now I want to let the user know where he clicked, so is it possible
>> to draw some sort of red border of a few pix around the clicked
>> draw-widget? I know they have a frame property, but that's just too thin
>> and doesn't attract attention...
>
> You could make the frame thicker. Something like:
>
> IDL> b=widget_base(/ROW)
> IDL> t=widget_draw(b,xsize=200,ysize=200,FRAME=10)
> IDL> b2=widget_base(b,xsize=200,ysize=200,xpad=10,ypad=10,/COLUMN)
> IDL> t2=widget_draw(b2,xsize=200,ysize=200)
> IDL> widget_control, b,/realize
>
> Unfortunately with this setup, you'll have to kill and re-create

> everything whenever the user selects a new draw widget (since FRAME's
> are only possible when the widget is created). Not ideal. You'll
> probably want to use UPDATE=0/1 to avoid flickering.
>
> Here's a trick that avoids all that killing/recreating. Use a
> bulletin board base (no /ROW or /COLUMN), and then layer two bases
> inside it: one which holds your draw widget, offset by 10 pixels in x an
> y, and one empty base which has FRAME=10 set, whose only purpose in life
> is to display that frame. Simply map and unmap the empty draw base to
> add/remove the frame as necessary. Something like:
>
> IDL> b=widget_base()
> IDL> b1=widget_base(b,xsize=200,ysize=200,xpad=10,ypad=10)
> IDL> t=widget_draw(b1,xsize=200,ysize=200)
> IDL> b2=widget_base(b,xsize=200,ysize=200,FRAME=10)
> IDL> widget_control, b,/realize
> IDL> widget_control, b2,map=0 ; remove frame
> IDL> widget_control, b2,map=1 & widget_control, b1,map=1 ; add frame back
>
> To make it easier, wrap this functionality up in a compound
> "frame-toggle-draw" widget of some sort, and then layout as many of
> these as you need. I might instead make it an object widget for
> additional convenience (so I can pass it an image to draw, tell it to
> erase, etc.), but a regular compound widget would work as well. Then
> something like:
>
> widget_control,frame_toggle_draw_widget_id,SET_VALUE=0 ; turn frame off
>
> could be enough to "de-select" that draw.
>
> JD
>
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...

Laurens

Subject: Re: border around draw widget
Posted by David Fanning on Wed, 27 Sep 2006 21:45:51 GMT
View Forum Message <> Reply to Message

Laurens writes:

> Could you tell me how to make such a widget-object? It sounds like
> something I was already thinking about...

Oh, I am SO glad I am working this week. I could hardly
resist this. :-)

Cheers,

David

P.S. If you don't understand JD's answer (believe me,
I only understand a fraction of them!), you could have
a look at something like FSC_Inputfield, which is a
sort of object/widget, albeit one that is too complicated
to serve as much of an example.

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

Subject: Re: border around draw widget
Posted by JD Smith on Wed, 27 Sep 2006 22:23:36 GMT
View Forum Message <> Reply to Message

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

Subject: Re: border around draw widget
Posted by Laurens on Thu, 28 Sep 2006 07:21:16 GMT
View Forum Message <> Reply to Message

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.

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.

Subject: Re: border around draw widget
Posted by Laurens on Wed, 04 Oct 2006 08:09:28 GMT
View Forum Message <> Reply to Message

Rick Towler wrote:
> 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.

Thanks very much! That was more about was I had in mind... since I only work once a week on this program I happened to have made nothing yet...

I use a class-object to store values like which one is selected. For what I understand, the
OPLOT, [0,0,1,1,0], [0,1,1,0,0], COLOR=255, THICK=4
line draws a rectangle around the object, but how do you remove it when its deselected? the "TV, thisData.image" command only draws the actual image, it seems.

Thanks again, and yeah, of course I'm using david's TVimage function :) in fact, the original TV function didn't work for me ;)

Subject: Re: border around draw widget
Posted by Laurens on Wed, 04 Oct 2006 09:18:53 GMT
View Forum Message <> Reply to Message

Rick Towler wrote:
> 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.
Never mind, it already works :) Thnx a bunch!

Subject: Re: border around draw widget
Posted by JD Smith on Mon, 09 Oct 2006 19:38:04 GMT
View Forum Message <> Reply to Message

On Thu, 28 Sep 2006 10:17:24 -0700, Rick Towler wrote:

> 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.

Yes, trivial, unless you need all pixels for drawing. Yes, you could just draw over the top of the image, assuming the loss of a few pixels around the edge is irrelevant. Otherwise, the base-on-base trick might be worth it, to keep your drawing code simple (...I have a canvas which is 248x248 pixels if selected, or 256x256 otherwise...).

JD
