Subject: Re: Widget to play and pause image stack display
Posted by Russell Ryan on Thu, 17 May 2012 14:51:40 GMT
View Forum Message <> Reply to Message

On May 11, 10:57 am, Helder <hel...@marchetto.de> wrote:
> On Thursday, May 10, 2012 6:40:11 PM UTC+2, Russell Ryan wrote:
>> On May 10, 12:35 pm, Russell Ryan <rr...@stsci.edu> wrote:
>>> On May 10, 10:24 am, Helder <hel...@marchetto.de> wrote:
>
>>>> Hi,
>>>> I would like to have a base widget with a draw widget where I play a movie (as in displaying a series of images from a stack).
>>>> I would like to have the typical two buttons that start the play of the movie and the pause button.
>>>> I'm not that good with widgets, but I can't thick of how I would be able to interrupt a loop that displays the images.
>>>> I would have had something like:
>
>>>> PauseButtonPressed = 0
>>>> ImgNr = 0
>>>> LoopInterval = 0.2
>>>> WHILE ~PauseButtonPressed THEN
>>>> TV, Image[ImgNr,*,*]
>>>> ImgNr++
>>>> IF ImgNr GT nImages THEN ImgNr = 0
>>>> WAIT, LoopInterval
>>>> ENDWHILE
>
>>>> Is there a way to check if the user has clicked something in the loop?
>
>>>> Thanks,
>>>> Helder
>
>>> Hi Helder
>>> Yeah, there is a way to do this....
>>> Here's a little example:
>
>>> pro timer_event,event
>>> widget_control,event.id,get_uval=uval
>
>>> case uval of
>>> 'START': begin
>>> widget_control,event.top,get_uval=state
>>> (*state).stop=0b
>>> widget_control,(*state).wtime,timer=(*state).time
>>> end
>>> 'RESTART': begin
>>> widget_control,event.top,get_uval=state

```
>>>       (*state).iter=0L
>>>       (*state).stop=0b
>>>       widget_control,(*state).wtime,timer=(*state).time
>>>     end
>>>     'STOP': begin
>>>       widget_control,event.top,get_uval=state
>>>       (*state).stop=1b
>>>     end
>>>     'TIME': begin
>>>       widget_control,event.top,get_uval=state
>
>>>       ;your movie stuff here:
>>>       n=50
>>>       x=findgen(n)/(n-1)*2*!PI
>>>       x+=(*state).iter*2*!PI/10.
>>>       plot,x,sin(x)
>>>       ;end of movie stuff
>
>>>       ;more to end the loop
>>>       if (*state).stop then return
>>>       if ++(*state).iter gt (*state).maxiter then return
>>>       widget_control,event.id,timer=(*state).time
>>>     end
>
>>>     'DRAW':
>>>     else:
>>>   endcase
>>>   end
>
>>>   pro timer
>
>>>   base=widget_base(/col)
>>>   wtime=widget_base(base,uval='TIME')
>>>   wdraw=widget_draw(base,xsize=200,ysize=200,uval='DRAW')
>>>   wstart=widget_button(base,value='Start',uval='START')
>>>   wrestart=widget_button(base,value='Restart',uval='RESTART')
>>>   wstop=widget_button(base,value='Stop',uval='STOP')
>
>>>   state={wdraw:wdraw,$
>>>          wtime:wtime,$
>>>          wstart:wstart,$
>>>          wstop:wstop,$
>>>          time:0.1,$
>>>          stop:0b,$
>>>          maxiter:100l,$
>>>          iter:0l}
>>>   state=ptr_new(state,/no_copy)
>>>   widget_control,base,/realize,set_uval=state
```

>>> xmanager,'timer',base,/no_block
>
>>> end
>
>> I should have said, the speed of the animation is set by time (which
>> is in seconds). I set it to 0.1 by default (see the state
>> structure). The looping variable is iter and the maximum size (which
>> you don't necessarily need) is set by maxiter. You could change this
>> to be max runtime and keep track of the length of time the animation
>> has run, but that might get you in to trouble (like if the CPU slows
>> down because you're doing something else). But the magic is in the
>> widget_timer. My organization of the event_handler is probably not
>> ideal, but it is just a get-you-started example.
>
>> Good Luck,
>> Russell
>
> Dear Russell,
> that's great and works just fine for me. I wasn't aware of the keyword uvalue for widgets (well actually I was aware, but I just didn't know what the this was for!).
> I see that this can be set just to manage such loops.
>
> Again, thanks a lot for your help. Really appreciated.
>
> Cheers, Helder

HI Helder,

Yeah, I wasn't sure what the uvalue was useful for on the widgets. I
took this approach after dissecting several GUIs written by ITT/RSI.
Though, after a bit of soul-searching, I wonder if it wouldn't be
better to use the uname keyword for this purpose, since you can set
the uvalue to any datatype and uname can only be a string. That way,
you can actually store some data (such as a structure or pointer) in
the uvalue.... Maybe someone can tell me why that's a foolish thing
to do...

R

PS. The way I organize the event handlers is to have a set of nested
case statements. The first will test which type of event it was:
WIDGET_BUTTON, WIDGET_DRAW, WIDGET_TEXT_DEL, etc. Then process each
widget within that case (with essentially this stuff). I feel like
that will be more efficient, since I find that for certain events
(like WIDGET_TEXT) I'm always needing to do a few things (like get the
state pointer), whereas with WIDGET_DRAW or WIDGET_BUTTON I don't
always need it... Plus, you can easily process things like resizing

the main window or closing the program (by the little red X in Windows or the red circle in OS X).