Subject: Re: Spin button widget??
Posted by Benjamin Luethi on Thu, 22 Sep 2005 09:24:41 GMT
View Forum Message <> Reply to Message

Hi,

Try WIDGET_SLIDER using the /VERTICAL and SCR_YSIZE keywords.

ben

On Thu, 22 Sep 2005 09:45:31 +0200, Antonio Santiago
<santiago@grahi.upc.edu> wrote:

> Hello group,
>
> anybody knows or had implemented a compound widget or object that acts
> as spin button, that is, a text with some numeric value and two little
> button to pop up/down its value.
>
> Thanks.
>

--
----------+

Subject: Re: Spin button widget??
Posted by David Alexander on Thu, 22 Sep 2005 15:22:50 GMT
View Forum Message <> Reply to Message

There's one in the IDL distribution, but it's not yet documented.
Still, you may be able to use it as the basis for what you need.

See <IDL_Distribution>/lib/itools/ui_widgets/cw_itupdownfield.pro

David

Subject: Re: Spin button widget??
Posted by b_gom on Thu, 22 Sep 2005 16:48:21 GMT
View Forum Message <> Reply to Message

Hi Antonio,

this might not be exactly what you're looking for, but this widget lets

you scroll through an arbitrary list of strings or numbers. Typing in the text field will select the nearest valid entry. The code is also on the RSI user contribution page.

Brad

```
; NAME:
;   BGSlider_Widget
;
; PURPOSE:
;
; A compound widget object for a slider that contains arbitrary items.
;
; CALLING SEQUENCE:
;
; widgetID = BGSlider_Widget(parent)
;
; INPUT PARAMETERS:
;
; parent -- The parent widget ID of the compound widget. Required.
;
; INPUT KEYWORDS:
;
; Event_Pro -- The event handler procedure for this compound widget.By
default: "".
; Event_Func -- The event handler function for this compound widget. By
default: "".
;
; If neither EVENT_PRO or EVENT_FUNC is defined, program events are
handled internally by the compound widget.
;
; UValue -- User value for any purpose.
; XSize -- The X size of the widget
; Editable -- Set to make the slider editable
; UValue -- The user value for any purpose.
; Values -- An array of string values corresponding to the slider
positions
; Start_index -- The initial position of the slider
; XSize -- The X size of the widget in characters.
; Scr_xsize -- The screen X size, in pixels
; Scr_ysize -- The screen Y size, in pixels
; Sensitive -- Set this keyword to control the initial sensitivity
state of the widget.
;
; OUTPUT KEYWORDS:
;
```

```
;   Obj_Ref -- Assign this keyword to an output variable that will hold
the internal object reference.
;               With the object reference you can call object methods
to easily change many properties of
;               the compound widget.
;
; EVENT STRUCTURE:
;
;   All events are handled internally unless either the Event_Pro or
Event_Func
;   keywords are used to assign an event handler to the compound
widget. All events
;   generated by the text widgets are passed to the assigned event
handler.
;
;   event = { BGSlider_Widget, $     ; The name of the event structure.
;             ID: 0L, $              ; The ID of the compound widget's
top-level base.
;             TOP: 0L, $             ; The widget ID of the top-level
base of the hierarchy.
;             HANDLER: 0L, $         ; The event handler ID. Filled out
by IDL.
;           }
;
; GETTING and SETTING VALUES:
;
;   So as not to disrupt the accepted paradigm in using compound
widgets, you
;   can use the return value of the function with WIDGET_CONTROL to
;   get and set the "value" of the widget.
;
;       Widget_Control, widgetID,
Set_Value=!root+'\IDL52\DATA\cyclone.dat'
;
;   Similarly, you can get the "value" of the widget:
;
;       Widget_Control, widgetID, Set_Value=theValue
;       Print, theValue
;
;
; USING OBJECT METHODS to CHANGE PROGRAM PROPERTIES:
;
;   If you obtain the object reference, you have a great deal more
control
;   over the properties of the compound widget. You obtain the object
reference
;   by calling the function like this:
;
```

```
;      widgetID = BGSlider_Widget(parent, ObjectRef=theObject)
;
; OBJECT PROCEDURE METHODS:
;
;   GetProperty -- This method allows various properties of the widget
to be
;       returned via output keywords. The keywords that are available
are:
;
;      Event_Func -- The name of the event handler function for this
compound widget.
;      Event_Pro -- The name of the event handler procedure for this
compound widget.
;      Parent -- The parent widget of the compound widget.
;      TLB -- The top-level base of the compound widget.
;      UValue -- The user value of the compound widget.
;      XSize -- The X size of the widget.
;
;   SetProperty -- This method allows various properties of the widget
to be
;       set via input keywords. The keywords that are available are:
;
;      Event_Func -- The name of the event handler function for this
compound widget.
;      Event_Pro -- The name of the event handler procedure for this
compound widget.
;      UValue -- The user value of the compound widget.
;      XSize -- The X size of the widget.
;
; SetValue, val, numeric=numeric -- this procedure sets the slider
position to the given string value, or
;     integer index value. If numeric is set, then val must be a
numeric value, and
;     the slider is set to the nearest matching value.
;
; SetAllValues,vals,index=index, format=format -- this procedure resets
the array of values
;    to vals, or the value at index to vals. Vals is a string or
numeric array or scalar.
;    If a numeric value is given, format specifies the string
formatting code to use.
;
; Increment -- increments the slider position by one
;
; Decrement -- decrements the slider position by one
;
;
; OBJECT PROCEDURE FUNCTIONS:
```

```
;
; GetTLB -- Returns the top-level base ID of the compound widget. No
Parameters.
;
; GetValue,index=index -- Returns the currently selected value. Set the
value
;  keyword to a variable to contain the current index
;
; GetIndex,last=last -- Returns the currently selected index. Set the
value
;  keyword to a variable to contain the last index
;
; GetAllValues -- Returns the full array of values
;
; MODIFICATION HISTORY:
;  Written by: Brad Gom, 1999.
; May 19, 2005 - Added sesitive keyword, fixed minor sizing bugs.
; May 24, 2005 - Added support for numeric values in the SetAllValues
and SetValue methods.
;-

FUNCTION BGSlider_Widget_Check_Values, values, format=format
; check the values array to be sure the type is acceptable. If it is,
update the object array.
 switch size(values,/type) of
  1:
  2:
  3:
  4:
  5:
  9:
  12:
  13:
  14:
  15:begin ;numeric
     values=strtrim(string(values,format=format))
     result=where(values eq '***',count)
     if count gt 0 then begin
      ok = Dialog_Message('BGSlider_Widget: Some values cannot be
displayed using this format code!.')
      RETURN, -1
    end
   numeric=1
   break
   end
  7: begin;string
   numeric=0
   break
```

```
   end
  else: begin ;not supported
     ok = Dialog_Message('BGSlider_Widget: The values must be string or
simple numeric type!.')
     RETURN, -1
   end
  endswitch

 return,numeric
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

FUNCTION BGSlider_Widget::GetTLB
 RETURN, self.tlb
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

PRO BGSlider_Widget::SetProperty, $
   Event_Func=event_func, $        ; The event handler function for
this compound widget.
   Event_Pro=event_pro, $          ; The event handler procedure for
this compound widget.
   UValue=uvalue, $                ; User value for any purpose.
   XSize=xsize                  ; The X size of the widget.

 Catch, theError
 IF theError NE 0 THEN BEGIN
    ok = Dialog_Message(!Err_String)
    RETURN
  ENDIF

 IF N_Elements(event_pro) NE 0 THEN BEGIN
    self.event_pro = event_pro
    Widget_Control, self.tlb, Event_Pro=event_pro
  ENDIF

 IF N_Elements(event_func) NE 0 THEN BEGIN
    self.event_func = event_func
    Widget_Control, self.tlb, Event_Func=event_func
  ENDIF

 IF N_Elements(uvalue) NE 0 THEN *self.uvalue = uvalue

 IF N_Elements(xsize) NE 0 THEN BEGIN
  self.xsize=xsize
  widget_control,self.textID, xsize=self.xsize
  text_xsize=(widget_info(self.textID,/geometry)).scr_xsize
  widget_control,self.sliderID, scr_xsize=text_xsize
```

```
  ENDIF

END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

PRO BGSlider_Widget::GetProperty, $
   Event_Func=event_func, $       ; The name of the event handler
function for this compound widget.
   Event_Pro=event_pro, $         ; The name of the event handler
procedure for this compound widget.
   Parent=parent, $               ; The parent widget of the compound
widget.
   TLB=tlb, $                     ; The top-level base of the compound
widget.
   UValue=uvalue, $               ; The user value of the compound
widget.
   XSize=xsize                    ; The X size of the widget.

 Catch, theError
 IF theError NE 0 THEN BEGIN
    ok = Dialog_Message(!Err_String)
    RETURN
 ENDIF

 event_pro = self.event_pro
 event_func = self.event_func
 parent=self.parent
 tlb = self.tlb
 uvalue = *self.uvalue
 xsize = self.xsize
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

PRO BGSlider_Widget::SetValue,val,numeric=numeric

 if n_elements(numeric) then begin ;treat the value as a number and
find the nearest match
  if self.numeric ne 1 then begin
   message,'The slider does not contain numeric values!',/info
   return
   endif
  result=min(abs(double(*self.values)-val[0]),ind)
  widget_control,self.textID,set_value=(*self.values)[ind]
  widget_control,self.sliderID,set_value=ind
  endif else begin
  ;treat the value as a string or index
  s=size(val,/type)
  if s eq 7 then begin  ;string
```

```
   ind=where(strtrim(*self.values) eq strtrim(val))
   if ind[0] ne -1 then begin
    widget_control,self.textID,set_value=(*self.values)[ind[0]]
    widget_control,self.sliderID,set_value=ind[0]
    endif
   endif else begin  ;index
   ind=long(val)
   widget_control,self.textID,set_value=(*self.values)[ind]
   widget_control,self.sliderID,set_value=ind
   endelse
  endelse
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

PRO  BGSlider_Widget::SetAllValues,vals,index=index,format=format
; if ptr_valid(self.values) then ptr_free,self.values ;unnecessary

 result = BGSlider_Widget_check_values(vals, format=format)
 if result eq -1 then return
 self.numeric = result
 *self.values=vals

 n=n_elements(vals)
 widget_control,self.sliderID,set_slider_max=n-1
 if n_elements(index) eq 0 then
widget_control,self.sliderID,get_value=index
 if index gt n-1 then begin
  index=n-1
  widget_control,self.sliderID,set_value=n-1
  endif
 widget_control,self.textID,set_value=(*self.values)[index]
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

PRO BGSlider_Widget::Increment
 widget_control,self.sliderID,get_value=ind
 ind=(ind+1)<n_elements(*self.values)
 widget_control,self.sliderID,send_event={WIDGET_SLIDER,
ID:self.sliderID, TOP:self.tlb,$
  HANDLER:self.tlb, VALUE:ind, DRAG:0}   ;**the handler should be set
properly..
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

PRO BGSlider_Widget::Decrement
 widget_control,self.sliderID,get_value=ind
 ind=(ind-1)>0
 widget_control,self.sliderID,send_event={WIDGET_SLIDER,
```

```
ID:self.sliderID, TOP:self.tlb,$
  HANDLER:self.tlb, VALUE:ind, DRAG:0}   ;**the handler should be set
properly..
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

FUNCTION BGSlider_Widget::GetValue,index=ind
 widget_control,self.textID,get_value=val
 widget_control,self.sliderID,get_value=ind
return,val[0]
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

FUNCTION BGSlider_Widget::GetIndex,last=last
 widget_control,self.sliderID,get_value=ind
 if ptr_valid(self.values) then last=n_elements(*self.values)-1
return,ind
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

FUNCTION BGSlider_Widget::GetAllValues
return,*(self.values)
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

PRO BGSlider_Widget_Set_Value, id, value
; This function sets a value of the widget, using the
; traditional "Widget_Control, fieldID, Set_Value=value" syntax.
 firstChild = Widget_Info(id, /Child)
 Widget_Control, firstChild, Get_UValue=self
 self->SetValue,value
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

FUNCTION BGSlider_Widget_Get_Value, id
; This function returns the value of the widget, using the
; traditional "Widget_Control, fieldID, Get_Value=value" syntax.
 firstChild = Widget_Info(id, /Child)
 Widget_Control, firstChild, Get_UValue=self
 RETURN, self->GetValue()
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------


;PRO BGSlider_ev_structure__Define
; The event structure definition. This is the event structure returned
by the compound widget.
;   eventStructure = { BGSlider_ev_structure, $  ; The name of the
```

```
event structure.
;                 ID: 0L,  $        ; The ID of the top-level base
of the compound widget.
;                 TOP: 0L, $        ; The ID of the top-level
base.
;                 HANDLER: 0L, $    ; The ID of the event handler
widget.
;                 Value: "" }   ; The value, if applicable.
;END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------


FUNCTION BGSlider_Widget_Event_Handler, event
; The main event handler for the compound widget. It reacts
; to "messages" in the UValues of widgets that generate events.
; The messages indicate which object method to call. A message
; consists of an object method and the self object reference.
 Widget_Control, event.ID, Get_UValue=theMessage
 self = theMessage.object

 if float(!VERSION.release) gt 5.0 then $
  Call_Method, theMessage.method, self, event $
  else Call_Procedure, 'BGSlider_Widget::'+theMessage.method, self,
event

   ; If an event handler routine has been set up for this
   ; compound widget, then an event will be created and sent.
 slider_event = {BGSlider_ev_structure, ID:self->GetTLB(),
TOP:event.top, HANDLER:0L, VALUE:self->GetValue() }

 self->GetProperty,event_pro=event_pro,event_func=event_func
 IF event_pro NE "" then begin
  call_procedure, event_pro, slider_event
  endif
 IF event_func NE "" then begin
  call_function, event_func, slider_event
  endif

 IF event_pro eq "" and event_func eq "" then
widget_control,self->GetTLB(),send_event=slider_event

RETURN, 0
END
 ;------------------------------------------------------------ ------------------------------------------------------------ ------

PRO BGSlider_Widget::Text_Events, event
 widget_control,self.sliderID,get_value=index
 widget_control,self.textID,get_value=val
```

```
 if self.numeric eq 0 then begin ;string values
  ind=where(*self.values eq val[0])
  if ind[0] ne -1 then begin
   widget_control,self.sliderID,set_value=ind[0]
   endif else
widget_control,self.textID,set_value=(*self.values)[index]
  endif else begin ;numeric values
   result=min(abs(double(*self.values)-double(val[0])),ind)
   widget_control,self.sliderID,set_value=ind
   widget_control,self.textID,set_value=(*self.values)[ind]
  endelse

;(*self.values)[index]=val
END
 ;--------------------------------------------------------- ------------------------------------------------------------ ------

PRO BGSlider_Widget::Slider_Events, event
 self->SetValue,event.value
END
 ;--------------------------------------------------------- ------------------------------------------------------------ ------

PRO BGSlider_Widget_Notify_Realize, ID
; When the compound widget is realized.
;Widget_Control, ID, Get_UValue=self
END
 ;--------------------------------------------------------- ------------------------------------------------------------ ------


PRO BGSlider_Widget_Kill_Notify, ID
; When the compound widget dies, destroy the self object.
 Widget_Control, ID, Get_UValue=self
 Obj_Destroy, self
END
 ;--------------------------------------------------------- ------------------------------------------------------------ ------

PRO BGSlider_Widget::CLEANUP
 Ptr_Free, self.uvalue
 Ptr_Free, self.values
END
 ;--------------------------------------------------------- ------------------------------------------------------------ ------

PRO BGSlider_Widget__Define
   objectClass = { BGSlider_Widget, $  ; The object class.
      parent: 0L, $    ; The ID of the parent widget.
      tlb: 0L, $     ; The ID of the top-level base widget for the
compound widget.
      baseID: 0L, $    ; The ID of the base widget for the compound
```

```
widget.
      textID: 0L,$    ; The ID of the slider value text widget
      sliderID: 0L,$    ; The ID of the slider widget
      event_pro: "", $   ; The user-defined name of the event procedure
for the widget.
      event_func: "", $   ; The user-defined name of the event function
for the widget.
      values: ptr_new(), $ ; Pointer to an array of strings
corresponding to the slider values
      numeric: 0L,$    ; flag to indicate slider contains numeric
values
      uvalue: Ptr_New(), $  ; The user's user value. Unused by the
program.
      xsize: 0L $    ; The X size of the widgets.
      }
END
 ;------------------------------------------------------------ ------------------------------------------------------------------ ------

FUNCTION BGSlider_Widget::INIT, $
   parent, $      ; The parent widget ID of the compound widget.
   Event_Pro=event_pro, $  ; The event handler procedure for this
compound widget.By default: "".
   Event_Func=event_func, $ ; The event handler function for this
compound widget. By default: "".
   editable=editable, $ ;
   format=format,$    ; Format string to convert values if they are
numeric
   UValue=uvalue, $    ; User value for any purpose.
   Values=values, $    ; The String values for the slider
   start_index=ind,$
   XSize=xsize,$      ; The X size of the widget.
 scr_xsize=scr_xsize,$
 scr_ysize=scr_ysize,$
 sensitive=sensitive,$
 obj_ref=obj_ref

; Catch, theError
; IF theError NE 0 THEN BEGIN
;    Catch, /Cancel
;    ok = Dialog_Message(!Err_String)
;    RETURN, 0
; ENDIF

   ; Populate self object.
 self.parent = parent
 self.event_pro = event_pro
 self.event_func = event_func
 self.uvalue = Ptr_New(uvalue)
```

```
 self.numeric = BGSlider_Widget_check_values(values, format=format)
 if self.numeric eq -1 then return,-1
 self.values = ptr_new(values)

 if n_elements(ind) eq 0 then ind=0
 if n_elements(xsize) eq 0 then self.xsize=max(strlen(values)) else
self.xsize=xsize

   ; Build widgets.
   ;base text box size on maximum string length or xsize keyword, in
that order
   ;base slider size on scr_xsize keyword, or text box size, in that
order

 self.tlb = Widget_Base( self.parent, Event_Func = self.event_func,
Event_Pro = self.event_pro, $
  Func_Get_Value='BGSlider_Widget_Get_Value',
Pro_Set_Value='BGSlider_Widget_Set_Value', $
  UValue=*self.uvalue ,space=0,xpad=0, sensitive=sensitive) ;store
'external' uvalue here.

  self.baseID = Widget_Base(self.tlb, UValue=self,/col,space=0,xpad=0,
 /base_align_center,kill_notify='BGSlider_Widget_Kill_Notify' ) ;store
object here

   self.textID = Widget_Text(self.baseID, Editable=editable, $
    Event_Func='BGSlider_Widget_Event_Handler',
Value=(*self.values)[ind], xsize=self.xsize,$
     YSize=1, UValue={method:'Text_Events', object:self} )
   text_xsize=(widget_info(self.textID,/geometry)).scr_xsize
   if n_elements(scr_xsize) eq 0 then scr_xsize=0
   slider_xsize=max([text_xsize,scr_xsize])

   self.sliderID = Widget_Slider(self.baseID,
Event_Func='BGSlider_Widget_Event_Handler',$
    Value=ind, /SUPPRESS_VALUE, scr_xsize=slider_xsize,
max=n_elements(values)-1,$
    UValue={method:'Slider_Events', object:self})

 if n_elements(scr_ysize) ne 0 then begin
  text_ysize=(widget_info(self.textID,/geometry)).scr_ysize
  widget_control,self.sliderID, scr_ysize=scr_ysize-text_ysize
  endif

 obj_ref=self
 RETURN, 1
END
```

```
;------------------------------------------------------------ ------------------------------------------------------------ ------


FUNCTION BGSlider_Widget, $
   parent, $      ; The parent widget ID of the compound widget.
   Event_Pro=event_pro, $  ; The event handler procedure for this
compound widget.By default: "".
   Event_Func=event_func, $ ; The event handler function for this
compound widget. By default: "".
   ObjectRef=objectref, $  ; An output keyword containing the object
reference.
   editable=editable,$
   format=format,$    ; Format string to convert values if they are
numeric
   UValue=uvalue, $    ; User value for any purpose.
   Values=values, $    ; The string values corresponding to the slider
positions
   start_index=ind,$
   XSize=xsize,$      ; The X size of the widget.
 scr_xsize=scr_xsize,$
 scr_ysize=scr_ysize,$
 sensitive=sensitive,$
 obj_ref=theObject

 Catch, theError
 ;theError = 0
 IF theError NE 0 THEN BEGIN
    ok = Dialog_Message(!Err_String)
    RETURN, 0
 ENDIF

   ; Need a parent parameter.
 IF N_Elements(parent) EQ 0 THEN BEGIN
    ok = Dialog_Message('BGSlider_Widget: A parent parameter is
required.')
    RETURN, -1
 ENDIF

   ; Check keyword arguments.

 IF N_Elements(event_pro) EQ 0 THEN event_pro = ""
 IF N_Elements(event_func) EQ 0 THEN event_func = ""
 IF N_Elements(uvalue) EQ 0 THEN uvalue = ""
 if n_elements(values) eq 0 then values =
string(indgen(101),format='(I3)')


   ; Create the underlying structure.
```

```
 obj = Obj_New('BGSlider_Widget', $
    parent, $
    Event_Pro=event_pro, $
    Event_Func=event_func, $
    editable=editable,$
    UValue=uvalue, $
    Values=values,$
    format=format,$
    start_index=ind,$
    XSize=xsize,$
    scr_xsize=scr_xsize,$
    scr_ysize=scr_ysize,$
  sensitive=sensitive,$
  obj_ref=o)

 theObject=o

   ; Return the ID of the top-level base of the compound widget.
  RETURN, obj->GetTLB()
END
 ;------------------------------------------------------------ ---------------------------------------------------------- ------


pro example_event,ev

 widget_control,ev.top, get_uvalue=info
 widget_control,ev.id, get_uvalue=uval
 case uval of
  'slider':begin
   print,'slider event returned: ',ev.value
   end
  'sens':begin
   widget_control,info.sliderID,sensitive=ev.select
   end
  'resize':begin
   info.sliderobj->setproperty,xsize=6
   end
  'set_all':begin
   info.sliderobj->setAllValues,[1,3,4.5,6,7.3],format='(f4.1)'
   end
  'set_val':begin
   info.sliderobj->setValue,4,/numeric
   end
  else:
  endcase

end
```

```
pro example

 base=widget_base(/col)

  values=findgen(1000)/10
  n= BGSlider_Widget(base,
 /editable,values=values,format='(F4.1)',uvalue='slider',scr_ xsize=100)
  values=['a','b','c','testing','1','2','3']
  n= BGSlider_Widget(base, /editable,values=values,uvalue='slider')
  values=['cat','dog','horse']
  s= BGSlider_Widget(base,
 Obj_Ref=sliderobj,sensitive=0,values=values,xsize=20,uvalue= 'slider')
  b=cw_bgroup(base,'Sensitive',/nonexclusive,uvalue='sens')
  b=widget_button(base,value='Resize',uvalue='resize')
  b=widget_button(base,value='Set All Values',uvalue='set_all')
  b=widget_button(base,value='Set Value to 4',uvalue='set_val')

 widget_control,base,/real

 info={tlb:base,$
  sliderID:s,$
  sliderobj:sliderobj}

 widget_control,base,set_uvalue=info

 xmanager,'example',base
end


-----------------------------------------



Antonio Santiago wrote:
>  Hello group,
>
>  anybody knows or had implemented a compound widget or object that acts
>  as spin button, that is, a text with some numeric value and two little
>  button to pop up/down its value.
>
>  Thanks.
>
>  --
>  ----------------------------------------------------
>  Antonio Santiago Pérez
>  ( email: santiago<<at>>grahi.upc.edu       )
>  (   www: http://www.grahi.upc.edu/santiago )
```

> (www: http://asantiago.blogsite.org)
> ---
> GRAHI - Grup de Recerca Aplicada en Hidrometeorologia
> Universitat Politècnica de Catalunya
> ---

Subject: Re: Spin button widget??
Posted by peter.albert@gmx.de on Fri, 23 Sep 2005 06:48:52 GMT
View Forum Message <> Reply to Message

> There's one in the IDL distribution, but it's not yet documented.
> Still, you may be able to use it as the basis for what you need.
>
> See <IDL_Distribution>/lib/itools/ui_widgets/cw_itupdownfield.pro
>
> David

Hi David,

thanks for the tip, I just built it into an application. However, I'd like to mention a trap I stepped into: if setting the INCREMENT keyword to a float value like 0.1, the arrow_up button seems not to work. If you set it to a double precision value, it works. arrow_down always works. Very weird.

Peter

Subject: Re: Spin button widget??
Posted by raval.chintan on Fri, 23 Sep 2005 14:28:00 GMT
View Forum Message <> Reply to Message

Hi Antonio,

You can find this program in IDL(6.1 or 6.2) / lib/ itools/ ui_widgets/ cw_itupdownfield.pro

Hope this can be help full.

With Regards
Chintan Raval.

Subject: Re: Spin button widget??
Posted by David Alexander on Fri, 23 Sep 2005 16:03:02 GMT

Peter,

I'm not sure why that would be, since when you set the INCREMENT keyword the routine will convert the value to a double before it does anything else. This is the first line of the routine:

```
increment = N_ELEMENTS(incrementIn) ? DOUBLE(incrementIn[0]) : 0.1d
```

Are you sure this is the cause of the faulty up arrow?

David

Subject: Re: Spin button widget??
Posted by edward.s.meinel@aero. on Fri, 23 Sep 2005 17:01:47 GMT

Peter Albert wrote:
>
> thanks for the tip, I just built it into an application. However, I'd
> like to mention a trap I stepped into: if setting the INCREMENT keyword
> to a float value like 0.1, the arrow_up button seems not to work. If
> you set it to a double precision value, it works. arrow_down always
> works. Very weird.
>
> Peter

Numerical analysis should be a required course for anyone who programs...

Computers use base 2 arithmetic, not base 10. Therefore 0.1 cannot be represented exactly on a computer. In particular, 0.1 in single precision IS NOT EQUAL TO 0.1 in double precision.

Subject: Re: Spin button widget??
Posted by peter.albert@gmx.de on Mon, 26 Sep 2005 09:26:17 GMT

Hi David,

well, Edwards is right with his comment, double(0.1) is not the same as 0.1d:

```
IDL> print, double(0.1), format = '(f20.18)'
0.100000001490116119
```

```
IDL> print, 0.1d, format = '(f20.18)'
0.100000000000000006
```

The problem in the code is the variable "invInc", calculated from "1d/state.increment". In my case invInc (more or less) equals 10 when the increment is set to 0.1d, while it is somewhat smaller than 10 in case of the increment being 0.1:

```
IDL> print, 1d / 0.1d, format = '(f10.7)'
10.0000000
IDL> print, 1d / double(0.1), format = '(f10.7)'
 9.9999999
```

This difference later makes the variable "integerized" different to LONG64(integerized) (used in the following IF statement), where "CEIL(integerized)" is used to round to the nearesf fraction.

Well, and this is where things go wrong, as this IF statement just takes care of cases where e.g. the starting value is something like "2.4 times the increment". In that case FLOOR and CEIL just work fine for the "up" and "down" event.

As a possible solution I would suggest to just count the multiples of the increment instead of the value itself and to increase or decrease just thie multiple: This can be done by adding a new tag to the "state" structure, which is at first filled with "ROUND(start_value / increment):

```
(in function cw_itupdownfield)
   state = {VALUE: value, $
      INCREMENT: increment, $
      SPINTIME: spinTime, $
      NNN: round(value / increment), $
      UNITS: units}
```

Then, the function "cw_itupdownfield_updown" can get rid of the IF statement (if (integerized eq LONG64(integerized))), and instead two lines of code like

```
      state.nnn = keyword_set(down) ? state.nnn - 1 : state.nnn + 1
      newvalue = state.nnn * state.increment
```

will do the trick.

Then the function "cw_itupdownfield_value" also needs one addiitional line (after line 180:)

```
   state.nnn = round(newvalue / state.increment)
```

I just gave it a quick check but it worked fine so far.

Best regards,

Peter