

---

Subject: Re: Bug swatters needed: CW\_SELECTAXES

Posted by [davidf](#) on Fri, 25 Sep 1998 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Doug Larson ([djl@loki.srl.caltech.edu](mailto:djl@loki.srl.caltech.edu)) writes:

- > I have been trying to get a CW for selecting axes to work using
- > WIDGET\_LIST etc.
- >
- > I am including a test driver called "test\_selectaxes"
- >
- > Problems:
- > 1) It does not report which index was selected for the Y or Z axes,
- > only X
- > 2) It only likes the first instance of CW\_SELECTAXES, the others
- > appear but do
- > not respond to clicks, in fact it wipes out.
- >
- > Before you all leave for the weekend, please help me debug! I have to
- > get an App
- > done, hopefully using this widget, over the weekend before the boss
- > returns.
- > He seems to think nice GUIs are a waste of energy! :)

Your basic problem here is a misunderstanding of NAMED structures. Once you execute a named structure definition statement (i.e.,

```
struct = {DAVID, age:35, handsome:'yes', children:afew}
```

you cannot change that definition in that IDL session. Thus, you cannot have a return value from your CW be defined differently in different circumstances.

Nor can you change how any of the fields are defined. For example, you can't do this if N\_Elements(many) > N\_Elements(afew):

```
struct.children = many
```

In fact, if you are going to have a structure field whose definition will be changing, you want a pointer in that field:

```
struct = {DAVID, age:35, handsome:'yes', children:Ptr_New(afew)}  
Print, 'Number of Children: ', *struct.children  
*struct.children = many
```

It seems to me that you want to know the axis (I presume here that 0 means X, 1 means Y, and 2 means Z), and the

item returned from that list widget. So I rewrote your CW\_SELECTAXES\_EVENT event handler so that the return value is defined like this:

```
ret = { SelectAxes_event, ID:parent, TOP:event.top, HANDLER:0L, $
      axis:state.selectList, index:event.index, $
      items:Ptr_New(selectNames) }
```

Where axis is the type of axis, index is the index of the selected item for that axis, and items is the list of items in the list. The selected item is (\*ret.items)[ret.index].

Here is working code with your example.

Cheers,

David

P.S. I also had to change the font names to get it to run on my Windows machine. :-)

-----  
David Fanning, Ph.D.  
Fanning Software Consulting  
E-Mail: davidf@dfanning.com  
Phone: 970-221-0438, Toll-Free Book Orders: 1-888-461-0155  
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>  
-----

```
;  
*****  
;+  
; NAME:  
; CW_SelectAxes  
;-  
;  
;
```

```
=====   
FUNCTION CW_SelectAxes_get_value, id  
; Recover the state of this compound widget  
stash = WIDGET_INFO(id, /CHILD)  
WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY  
  
if(state.dimension eq 2)then $  
  ret = [ state.ListIndexX, state.ListIndexY ]  
  
if(state.dimension eq 3)then $  
  ret = [ state.ListIndexX, state.ListIndexY, state.ListIndexZ ]
```

```

WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY
return, ret
end
; -----
FUNCTION CW_SelectAxes_set_value, id
; Recover the state of this compound widget
stash = WIDGET_INFO(id, /CHILD)
WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY

if(state.selectList eq 0)then begin
state.ListIndexX = value
endif

if(state.selectList eq 1)then begin
state.ListIndexY = value
endif

if(state.selectList eq 2)then begin
state.ListNamesZ = value
endif

WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY
end
; -----
FUNCTION CW_SelectAxes_event, event

; Retrieve the structure from the child that contains the sub ids
parent=event.handler
stash = WIDGET_INFO(parent, /CHILD)
WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY
eventValue = event.index
;print, state.xaxisid, state.yaxisid, state.zaxisid, event.id
; See which list was clicked, the X, Y, or Z axis.
CASE event.ID OF
state.xaxisID : BEGIN
state.selectList = 0
selectNames = state.listNamesX
if(state.bugmsg eq 1)then print,'Xaxis List=',eventValue
END
state.yaxisID : BEGIN
state.selectList = 1
selectNames = state.listNamesY
if(state.bugmsg eq 1)then print,'Yaxis List=',eventValue
END
state.zaxisID : BEGIN
state.selectList = 2
selectNames = state.listNamesZ

```

```

        if(state.bugmsg eq 1)then print,'Zaxis List=',eventValue
        END
    ENDCASE

ret = { SelectAxes_event, ID:parent, TOP:event.top, HANDLER:0L, $
        axis:state.selectList, index:event.index,
items:Ptr_New(selectNames) }

WIDGET_CONTROL, stash, SET_UVALUE = state, /NO_COPY

RETURN, ret

END
; -----
-----
FUNCTION CW_SelectAxes, parent, DEBUG=bugmsg, DIMENSION = dimension, $
    LABELLIST=labellist, LABELFONT=labelfont, $
    LABELPLACE=LabelPlace, $
    TITLEAS = titles, TITLEFONT=titlefont, TITLEPLACE=titleplace, $
    FRAME = frame, UVALUE = uval, VALUE = val, $
    LISTNAMES=listnames, LISTFONT=listfont, LISTPLACE=listplace, $
    XAXISLIST = ListNamesX, $
    YAXISLIST = ListNamesY, $
    ZAXISLIST = ListNamesZ, $
    XSIZE = xsize, YSIZE = ysize, ZSIZE = zsize

if(N_PARAMS() EQ 0) then MESSAGE, 'Incorrect number of arguments'
ON_ERROR, 2 ;return to caller

; Defaults for keywords
; -----
if NOT(KEYWORD_SET(bugmsg)) then bugmsg = 0
if N_ELEMENTS(dimension) EQ 0 then dimension = 2
if N_ELEMENTS(labellist) EQ 0 then begin
    CASE dimension OF
    2: begin
        labellist = STRARR(2)
        labellist[0] = 'X Axis'
        labellist[1] = 'Y Axis'
        end
    3: begin
        labellist = STRARR(3)
        labellist[0] = 'X Axis'
        labellist[1] = 'Y Axis'
        labellist[2] = 'Z Axis'
        end
    ENDCASE
endif

```

```

if NOT(KEYWORD_SET(labelfont)) then labelfont = $
    'times'
if N_ELEMENTS(LabelPlace) EQ 0 then LabelPlace = 0
if N_ELEMENTS(titleas) EQ 0 then titleas = ""
if N_ELEMENTS(titlefont) EQ 0 then titlefont = labelfont
if N_ELEMENTS(titleplace) EQ 0 then titleplace = 0
if N_ELEMENTS(frame) EQ 0 then frame = 0
if N_ELEMENTS(uval) EQ 0 then uval = 0
if N_ELEMENTS(val) EQ 0 then val = 0
if N_ELEMENTS(xsize) EQ 0 then xsize = 6
if N_ELEMENTS(ysize) EQ 0 then ysize = 6
if N_ELEMENTS(listfont) EQ 0 then listfont = labelfont
if N_ELEMENTS(listplace) EQ 0 then listplace = 0

```

; The actual list elements can be specified on a per list  
; basis, or once with the LISTNAMES keyword.

```

nListNames = N_ELEMENTS(ListNames)
nListNamesX = N_ELEMENTS(ListNamesX)
nListNamesY = N_ELEMENTS(ListNamesY)
nListNamesZ = N_ELEMENTS(ListNamesZ)
if(nListNames eq 0)then begin
    filldefaults = 0L
    CASE dimension OF
        2: begin
            if((nListNamesX EQ 0) or (nListNamesY EQ 0)) then $
filldefaults = 1L
            end
        3: begin
            if((nListNamesX EQ 0) or $
                (nListNamesY EQ 0) or $
                (nListNamesZ EQ 0)) then begin
                filldefaults = 1L
            endif
        end
    ENDCASE
    if(filldefaults EQ 1L) then begin
        nListNamesX = 10
        nListNamesY = nListNamesX
        nListNamesZ = nListNamesX
        ListNamesX = STRARR(nListNamesX)
        ListNamesY = STRARR(nListNamesY)
        ListNamesZ = STRARR(nListNamesZ)
        for I = 0, nListNamesX-1 do begin
            name = 'Value '+strcompress(STRING(I))
            ListNamesX[I] = name
            ListNamesY[I] = name
            ListNamesZ[I] = name
        endfor
    end

```

```

endif
endif else begin
  nListNamesX = nListNames
  nListNamesY = nListNames
  nListNamesZ = nListNames
  ListNamesX = STRARR(nListNames)
  ListNamesY = STRARR(nListNames)
  ListNamesZ = STRARR(nListNames)
  ListNamesX = ListNames[*]
  ListNamesY = ListNames[*]
  ListNamesZ = ListNames[*]
endelse

; Setup placement of a title for this CW
; -----
if(titleplace eq 0)then begin      ; COLUMN
  SelectAxesTLB=WIDGET_BASE(parent, /COLUMN)
  titleBase = WIDGET_BASE(SelectAxesTLB, /ALIGN_CENTER)
endif else begin                  ; ROW
  SelectAxesTLB=WIDGET_BASE(parent, /ROW)
  titleBase = WIDGET_BASE(SelectAxesTLB, /ALIGN_CENTER)
endelse
title = WIDGET_LABEL(titleBase, VALUE=titles, FONT=titlefont)

; Setup layout of lists as a row or a column
; -----
if(listplace eq 0)then begin      ; ROW
  SelectAxesBase=WIDGET_BASE(SelectAxesTLB, /ROW)

; Setup placement of each individual list's title
; -----
CASE LabelPlace OF
  0 : BEGIN
    xaxisBase = WIDGET_BASE(SelectAxesBase, $
GROUP_LEADER = parent, /COL)
    labelxBASE = WIDGET_BASE(xaxisBase, /ALIGN_CENTER, /COL)
    labelx = WIDGET_LABEL(labelxBASE, VALUE=labellist[0], $
      FONT=labelfont)
    xaxisID = WIDGET_LIST(xaxisBase, VALUE=ListNamesX, $
      UVALUE=labellist[0], $
      FONT=listfont, $
      XSIZE=xsize, YSIZE=ysize)

    yaxisBase = WIDGET_BASE(SelectAxesBase, $
GROUP_LEADER = parent, /COL)
    labelyBase = WIDGET_BASE(yaxisBase, /ALIGN_CENTER, /COL)
    labely = WIDGET_LABEL(labelyBase, VALUE=labellist[1], $
      FONT=labelfont)

```

```

yaxisID = WIDGET_LIST(yaxisBase, VALUE=ListNamesY, $
    UVALUE=labellist[1], $
    FONT=listfont, $
    XSIZE=xsize, YSIZE=ysize)

if(dimension eq 3)then begin
    zaxisBase = WIDGET_BASE(SelectAxesBase, $
        GROUP_LEADER = parent, /COL)
    labelzbase = WIDGET_BASE(zaxisBase, /ALIGN_CENTER, /COL)
    labelz = WIDGET_LABEL(labelzbase, VALUE=labellist[2], $
        FONT=labelfont)
    zaxisID = WIDGET_LIST(zaxisBase, VALUE=ListNamesZ, $
        UVALUE=labellist[2], $
        FONT=listfont, $
        XSIZE=xsize, YSIZE=ysize)

endif else begin
    zaxisID = 0L
endif
END
1 : BEGIN
    xaxisBase = WIDGET_BASE(SelectAxesBase, $
GROUP_LEADER = parent, /ROW)
    labelxBASE = WIDGET_BASE(xaxisBase, /ALIGN_CENTER)
    labelx = WIDGET_LABEL(labelxBASE, VALUE=labellist[0], $
        FONT=labelfont)
    xaxisID = WIDGET_LIST(xaxisBase, VALUE=ListNamesX, $
        UVALUE=labellist[0], $
        FONT=listfont, $
        XSIZE=xsize, YSIZE=ysize)

    yaxisBase = WIDGET_BASE(SelectAxesBase, $
GROUP_LEADER = parent, /ROW)
    labelyBase = WIDGET_BASE(yaxisBase, /ALIGN_CENTER)
    labely = WIDGET_LABEL(labelyBase, VALUE=labellist[1], $
        FONT=labelfont)
    yaxisID = WIDGET_LIST(yaxisBase, VALUE=ListNamesY, $
        UVALUE=labellist[1], $
        FONT=listfont, $
        XSIZE=xsize, YSIZE=ysize)

if(dimension eq 3)then begin
    zaxisBase = WIDGET_BASE(SelectAxesBase, $
        GROUP_LEADER = parent, /ROW)
    labelzbase = WIDGET_BASE(zaxisBase, /ALIGN_CENTER)
    labelz = WIDGET_LABEL(labelzbase, VALUE=labellist[2], $
        FONT=labelfont)
    zaxisID = WIDGET_LIST(zaxisBase, VALUE=ListNamesZ, $

```

```

        UVALUE=labellist[2], $
        FONT=listfont, $
        XSIZE=xsize, YSIZE=ysize)
    endif else begin
        zaxisID = 0L
    endelse
END
ENDCASE
endif else begin          ; COLUMN
    SelectAxesBase=WIDGET_BASE(SelectAxesTLB, /COLUMN)

; Setup placement of each individual list's title
; -----
CASE LabelPlace OF
    0 : BEGIN
        xaxisBase = WIDGET_BASE(SelectAxesBase, $
GROUP_LEADER = parent, /COL)
        labelxBASE = WIDGET_BASE(xaxisBase, /ALIGN_CENTER, /COL)
        labelx = WIDGET_LABEL(labelxBASE, VALUE=labellist[0], $
            FONT=labelfont)
        xaxisID = WIDGET_LIST(xaxisBase, VALUE=ListNamesX, $
            UVALUE=labellist[0], $
            FONT=listfont, $
            XSIZE=xsize, YSIZE=ysize)

        yaxisBase = WIDGET_BASE(SelectAxesBase, $
GROUP_LEADER = parent, /COL)
        labelyBase = WIDGET_BASE(yaxisBase, /ALIGN_CENTER, /COL)
        labely = WIDGET_LABEL(labelyBase, VALUE=labellist[1], $
            FONT=labelfont)
        yaxisID = WIDGET_LIST(yaxisBase, VALUE=ListNamesY, $
            UVALUE=labellist[1], $
            FONT=listfont, $
            XSIZE=xsize, YSIZE=ysize)

        if(dimension eq 3)then begin
            zaxisBase = WIDGET_BASE(SelectAxesBase, $
                GROUP_LEADER = parent, /COL)
            labelzbase = WIDGET_BASE(zaxisBase, /ALIGN_CENTER, /COL)
            labelz = WIDGET_LABEL(labelzbase, VALUE=labellist[2], $
                FONT=labelfont)
            zaxisID = WIDGET_LIST(zaxisBase, VALUE=ListNamesZ, $
                UVALUE=labellist[2], $
                FONT=listfont, $
                XSIZE=xsize, YSIZE=ysize)
        endif else begin
            zaxisID = 0L
        endelse
    end
end

```



```

END
1 : BEGIN
  xaxisBase = WIDGET_BASE(SelectAxesBase, $
GROUP_LEADER = parent, /ROW)
  xaxisID = WIDGET_LIST(xaxisBase, VALUE=ListNamesX, $
    UVALUE=labellist[0], $
    FONT=listfont, $
    XSIZE=xsize, YSIZE=ysize)
  labelxBASE = WIDGET_BASE(xaxisBase, /ALIGN_CENTER)
  labelx = WIDGET_LABEL(labelxBASE, VALUE=labellist[0], $
    FONT=labelfont)

  yaxisBase = WIDGET_BASE(SelectAxesBase, $
GROUP_LEADER = parent, /ROW)
  yaxisID = WIDGET_LIST(yaxisBase, VALUE=ListNamesY, $
    UVALUE=labellist[1], $
    FONT=listfont, $
    XSIZE=xsize, YSIZE=ysize)
  labelyBase = WIDGET_BASE(yaxisBase, /ALIGN_CENTER)
  labely = WIDGET_LABEL(labelyBase, VALUE=labellist[1], $
    FONT=labelfont)

  if(dimension eq 3)then begin
    zaxisBase = WIDGET_BASE(SelectAxesBase, $
      GROUP_LEADER = parent, /COLUMN)
    zaxisID = WIDGET_LIST(zaxisBase, VALUE=ListNamesZ, $
      UVALUE=labellist[2], $
      FONT=listfont, $
      XSIZE=xsize, YSIZE=ysize)
    labelzbase = WIDGET_BASE(zaxisBase, /ALIGN_CENTER)
    labelz = WIDGET_LABEL(labelzbase, VALUE=labellist[2], $
      FONT=labelfont)
  endif else begin
    zaxisID = 0L
  endelse
END
ENDCASE
endelse

; The state is dependent on the number of axes
; -----
selectList=0
CASE dimension OF
  2: begin
    state = { SelectAxesTLB: SelectAxesTLB, $
      selectList: selectList, $
      bugmsg: bugmsg, $
      dimension: dimension, $

```

```

        xaxisID: xaxisID, $
        yaxisID: yaxisID, $
        ListIndexX: LONARR(nListNamesX), $
        ListIndexY: LONARR(nListNamesY), $
        ListNamesX: ListNamesX, $
        ListNamesY: ListNamesY }
    end
3: begin
    state = { SelectAxesTLB: SelectAxesTLB, $
        selectList: selectList, $
        bugmsg: bugmsg, $
        dimension: dimension, $
        xaxisID: xaxisID, $
        yaxisID: yaxisID, $
        zaxisID: zaxisID, $
        ListIndexX: LONARR(nListNamesX), $
        ListIndexY: LONARR(nListNamesY), $
        ListIndexZ: LONARR(nListNamesZ), $
        ListNamesX: ListNamesX, $
        ListNamesY: ListNamesY, $
        ListNamesZ: ListNamesZ }
    end
ENDCASE

```

```

WIDGET_CONTROL, SelectAxesTLB, SET_UVALUE = uval, $
    EVENT_FUNC = 'CW_SelectAxes_event', $
    PRO_SET_VALUE = 'CW_SelectAxes_set_value', $
    FUNC_GET_VALUE = 'CW_SelectAxes_get_value'

```

```

WIDGET_CONTROL, WIDGET_INFO(SelectAxesTLB, /CHILD), SET_UVALUE=state,
/NO_COPY
RETURN, SelectAxesTLB

```

```

END

```

```

;
; ***** UNIT TESTER FOR :: CW_SelectAxes
;
; *****
;
; *****
;+
; NAME: test_CW_SelectAxes_event
;
; PURPOSE:
; This is just a simple event loop for the unit tester. It demonstrates
; some of the features of this compound widget.
; -
;
;

```

```

=====
PRO test_CW_SelectAxes_event, event

;Help, event, /Structure

Print, 'Axis: ', event.axis
Print, 'Index: ', event.index
items = *event.items
Print, 'Selected Item: ', items[event.index]
END
; -----
;+
; NAME: test_CW_SelectAxes
;
PRO test_SelectAxes
sfont1 = 'times'
sfont2 = 'times'
sfont3 = 'times'

slideTLB=WIDGET_BASE(/col)
ListSetID0=CW_SelectAxes(slideTLB)
ListSetID1=CW_SelectAxes(slideTLB, DIMENSION = 3, UVALUE=1)

timelist=['Time', 'livetime']
detectorlist=['m1', 'm2', 'm3', 'adcor', 'D1', 'D2', 'D3', 'D4']
ListSetID2=CW_SelectAxes(slideTLB, DEBUG=1, $
    TITLEAS = 'Real Data', TITLEPLACE=0, $
    LABELFONT=sfont1, LABELPLACE = 0, $
    XAXISLIST = timelist, $
    YAXISLIST = detectorlist, $
    LISTPLACE=0, $
    UVALUE=2)

spkids=['Kenny', 'Kyle', 'Cartman', 'Stan', 'Damien', 'Ike', 'Shelley']
spadults=['Chef', 'Matt Stone', 'Trey Parker', 'Mr. Garrison', 'Ned', $
    'Officer Barbrady', 'Mr. Mackey', 'Big Gay Al', 'Miss
Crabtree']
spmisc=['Cows', 'Mr. Hankey', 'Mr. Hat', 'Terrance & Phillip']
ListSetID3=CW_SelectAxes(slideTLB, DEBUG=1, $
    TITLEAS = 'South Park Characters', TITLEPLACE=0, $
    LABELFONT=sfont1, LABELPLACE = 0, $
    LABELLIST = ['Kids', 'Adults', 'Oddities'], LISTPLACE=0, $
    DIMENSION = 3, $
    XAXISLIST = spkids, $
    YAXISLIST = spadults, $
    ZAXISLIST = spmisc, $
    UVALUE=3)

```

```
; Realize the widget:  
WIDGET_CONTROL, slideTLB, /REALIZE  
  
; Hand off control of the widget to the XMANAGER:  
XMANAGER, "test_CW_SelectAxes", slideTLB, $  
GROUP_LEADER=slideTLB, /NO_BLOCK
```

```
END
```

---