Subject: globalization of variables
Posted by taejon on Mon, 06 Nov 2006 14:13:50 GMT
View Forum Message <> Reply to Message

Hi

in the code below I want to use the variable voxel_a0 globally. That means, when I try after compilation (successful) to run the program I got the message:

variable is undefined : voxel_a0

In principle I want that after starting the program after pressing the 'apply_button' (eventhandler=applydata_event) the variables should be stored globally in the structure pState, so that when proceeding the program in the subprogram efgalc1_computedim I want to use them again, and later on in other functions or subprograms as well. Any idea ?

Thanks in advance

Sven Ohmann

```
;************************************************************ ********************************
pro efgcalc1_event, event
          widget_control, event.top, get_uvalue=pstate
          widget_control, event.id, Get_Value = buttonValue
print, buttonvalue
  case buttonValue of
         'QuitSofort'      : widget_control, event.top, /destroy
         'OPTIONS'         : efgcalc1_options_events, event
         'Load Binaryfile' : LoadBinFile_events, event
         'Dismis'          : dismisdata_event, event

         'Apply'           : Applydata_event, event
  endcase
end


;************************************************************ ********************************
; Wie im eventhandler 'loadbinfile_events' beschrieben (jetzt als
Funktion ausgelagert)
FUNCTION efgcalc1_loadbindata, inputfile, gs
  print, "Voxelzahl", gs
  data = ptr_new(dblarr(gs, gs, gs))
  openr, 1, inputfile
  readu, 1, *data
  close, 1
```

```
  print, 'max:', max(*data)
  print, 'min:', min(*data)
  return, data
end

 ;******************************************************** ******************************

pro applydata_event, event
  widget_control, event.top, get_uvalue=pstate
  print, 'Data applied'

  widget_control, (*pstate).ccaId,          get_value=a0
  widget_control, (*pstate).ccbId,          get_value=b0
  widget_control, (*pstate).cccId,          get_value=c0
  widget_control, (*pstate).ccalphaId,      get_value=alpha
  widget_control, (*pstate).ccbetaId,       get_value=beta
  widget_control, (*pstate).ccgammaId,      get_value=gamma
  widget_control, (*pstate).a0valId,        get_value=voxel_a0
  widget_control, (*pstate).b0valId,        get_value=voxel_b0
  widget_control, (*pstate).c0valId,        get_value=voxel_c0
  widget_control, (*pState).voxelzahlId,    get_value=gs

; Hier das urspruengliche Voxelvolumen
  Voxelvol = voxel_a0 * voxel_b0 * voxel_c0
  print, 'Voxelvolumen:', voxelvol

; Hier werden die Voxelkantenlaengen des Wuerfels auf array-ticks
korrigiert
;  voxel_a0 = a0/gs
;  voxel_b0 = b0/gs
;  voxel_c0 = c0/gs

;  print, 'a0-Werte:', voxel_a0
;  print, 'b0-Werte:', voxel_b0
;  print, 'c0-Werte:', voxel_c0
end
 ;******************************************************** ******************************
pro efgcalc1_Computedim, pState
; Hier werden die Voxelkantenlaengen des Wuerfels auf array-ticks
korrigiert
  widget_control, (*pState).voxelzahlId,    get_value=gs
  voxel_a0 = a0/gs
  voxel_b0 = b0/gs
  voxel_c0 = c0/gs

 print, 'a0-Werte:', voxel_a0
  print, 'b0-Werte:', voxel_b0
```

```
  print, 'c0-Werte:', voxel_c0
end

;
 ************************************************************ ******************************
; Funktion zum Umrechnen von Angstroem-Einheiten auf array-ticks
; *pState.ccaId gibt z.B. die reale Kristallaenge an (4.8215
Angstroem). *pState.a0val
; die Laenge in Voxellaengeneinheiten

;pro efgcalc1_computedim, pState
;print, 'ccaid:',(*pstate).ccaId
;print, 'Versuch', (*pState).ccaId
; widget_control, (*pState).voxelzahlId,    get_value=gs
;print , 'Die Voxelzahl pro Interval:', gs

;end
 ;************************************************************ ******************************

pro LoadBinFile_events, event
; Hier kann User die von EVOX bereitgestellte Datei auswaehlen !
  inputfile = DIALOG_PICKFILE(/READ, FILTER="*.bin")

; Falls keine Datei ausgewaehlt wurde...
  if (inputfile eq '') then begin $
     widget_control, event.top, SET_UVALUE=pState, /No_Copy
     RETURN
  endif

; Falls eine Datei ausgewaehlt wurde, wird aus dem Feld Voxelzahl die
; frueher bezeichnete Gridsize gewonnen und die Datei eingelesen. Es
erfolgt eine Pruefung
; der Min und Max-Werte ob dies erfolgreich geschah

  widget_control, event.top, get_uvalue=pstate
  widget_control, (*pstate).VoxelzahlId, get_value = gs

; Funktionsaufruf zum Einlesen der Binaerdatei
  (*pstate).maindata = efgcalc1_loadbindata(inputfile, gs)

; Funktionsaufruf um von Angstroem auf array--ticks umzurechnen
  efgcalc1_computedim, pState

end


 ;************************************************************ ******************************
pro efgcalc1
```

```
; Hauptprogramm, hier wird die graphische Oberfläche gebastelt
; Zuerst das Menue oben (File und Options), durch Menu=1 wird 'File' zu
einem pulldown-menue
; Mit Menu = 1 wird Button zu Pulldownmenu

  BaseId           = widget_base(/row, title=' EFG-Berechnung',
mbar=menubaseId)
  FileId           = widget_button(menubaseId, Value = 'File',    Menu
= 1)
  OptionsId        = widget_button(menubaseId, Value = 'Options', Menu
= 1)
  Q_Id             = widget_button(menubaseId, Value = 'Quit',    Menu
= 1)

; Hier unter dem 'File_Button', Event_pro gibt den Eventhandler an
  BinaryfileId     = widget_button(fileId,     Value = 'Load
Binaryfile', Event_Pro=efgcalc1_LoadBinFile_events)
  AsciifileId      = widget_button(fileId,     Value = 'Load
Asciifile')
  CoreBinfileId    = widget_button(fileId,     Value = 'Load Binary
Corefile')
  CoreAsciifileId  = widget_button(fileId,     Value = 'Load Ascii
Corefile')

; Hier unter 'Quit-Button'
  QuitId           = widget_button(Q_Id,       Value = 'QuitSofort')

; Hier unter dem 'Optionsbutton'
  MultiselectId    = widget_button(optionsId,  Value = 'Multiselect')
  ComputeEFGId     = widget_button(optionsId,  Value = 'Compute the
EFG')
  DrawEFGId        = widget_button(optionsId,  Value = 'Draw the EFG')

  SubbaseId        = widget_base(baseId, /col)

; Die Gruppe zum auswaehlen von 'settings' und 'tools'
; wTabselId        = CW_BGROUP(subbaseId, ['Settings', 'Tools'])

; Das Feld mit dem Titel 'Voxelzahl' (CW_Field ist fertiges Widget in
IDL), sowie Kristalldaten
  VoxelzahlId      = CW_Field(SubbaseId, Title = 'Voxelzahl', Value =
100, xsize =5, /Integer)

  KristalllabelId  = widget_label(subbaseId, Value='Kristallsystem')
  ccaId            = CW_Field(SubbaseId, Title = 'a0',
value = 4.8195, xsize=5, /float)
  ccbId            = CW_Field(SubbaseId, Title = 'b0',
value = 10.480, xsize=5, /float)
```

```
  cccId            = CW_Field(SubbaseId, Title = 'c0',
value = 6.0902, xsize=5, /float)
  ccalphaId         = CW_Field(SubbaseId, Title = 'alpha',
value = 90.0,   xsize=5, /float)
  ccbetaId          = CW_Field(SubbaseId, Title = 'beta',
value = 90.0,   xsize=5, /float)
  ccgammaId         = CW_Field(SubbaseId, Title = 'gamma',
value = 90.0,   xsize=5, /float)
  a0valId           = CW_Field(SubbaseId, Title = 'a0voxellength',
value = 0.05,   xsize=5, /float)
  b0valId           = CW_Field(SubbaseId, Title = 'b0voxellength',
value = 0.05,   xsize=5, /float)
  c0valId           = CW_Field(SubbaseId, Title = 'c0voxellength',
value = 0.05,   xsize=5, /float)

; Buttons zum Aufnehmen von Voxelzahl und Kristalldaten
  dismisId          = widget_button(subbaseId , Value = 'Dismis',
xsize=30, ysize=30)
  applyId           = widget_button(subbaseId,  Value = 'Apply',
xsize=30, ysize=30)

; Hier das Bild wo die Elektronendichte erscheinen soll
  DrawbaseId        = widget_base(baseId, /col)
  printId           = widget_base(drawbaseId, /col)
  DrawId            = widget_draw(printId, xsize=500, ysize=400)

; Hier die Schieberegler fuer das Rendern sowie Texteingabe der
d-Elektronenfile
  renderbaseId      = widget_base(printId, /row)
  RenderId          = widget_slider(renderbaseId, Title='render',
minimum=0, maximum=200)
  rendertext        = widget_text(renderbaseId, /editable, ysize=2)
  renderrefresh     = widget_button(renderbaseId, Value='Refresh')

; Hier dasselbe fuer das kugelsymmetrische Fc-File
  rendercorebaseId  = widget_base(printId, /row)
  RenderCore        = widget_slider(rendercorebaseId, Title='render
Core data', minimum=0, maximum=200)
  rendercoretext    = widget_text(rendercorebaseId, /editable, ysize=2)
  rendcorerefreshID = widget_button(rendercorebaseId,
Value='rendercorerefresh')


; Hier werden die Infos fuer IDL bereitgestellt. Konzept Siehe Fanning
S. 154

;  widget_control, baseId, /realize
  widget_control, drawId, get_value=winvis
```

```
  state  = {winvis : winvis, voxelzahlId : voxelzahlID, ccaId : ccaId,
ccbId:ccbId, cccId:cccId, $
          ccalphaId:ccalphaId, ccbetaId:ccbetaId,
ccgammaId:ccgammaId, $
          a0valId:a0valId, b0valId:b0valId, c0valId:c0valId, maindata
: ptr_new(/allocate_heap), $
          voxel_a0 : voxel_a0, voxel_b0 : voxel_bo, voxel_c0 :
voxel_c0, gs:gs }

  pstate = ptr_new(state)

  widget_control, baseId, Set_UValue=pstate
   widget_control, baseId, /realize
; Xmanager gibt den Eventhandler an, und welches das Hauptprogramm ist
  xmanager, 'efgcalc1', baseId, Event_Handler = 'efgcalc1_event'
,/no_block

end
 ;;****************************************************** ***************

pro dismisdata_event, event
  print, 'Data canceled'
; Setze die Textfelder fuer die Kristalldaten auf Null

  widget_control, event.top, get_uvalue=pstate
  widget_control, (*pstate).ccaId,         set_value='0'
  widget_control, (*pstate).ccbId,         set_value='0'
  widget_control, (*pstate).cccId,         set_value='0'
  widget_control, (*pstate).ccalphaId,     set_value='0'
  widget_control, (*pstate).ccbetaId,      set_value='0'
  widget_control, (*pstate).ccgammaId,     set_value='0'


end
 ;****************************************************** ********************************
```