

---

Subject: Re: Similar to a Widget\_Container  
Posted by [David Fanning](#) on Fri, 27 Feb 2004 13:24:03 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Antonio Santiago writes:

> Hi, I just finish an object 'boxtable'.  
>  
> With it you can create a widget\_base and an imaginri table and then you  
> can put another widget only specifying de cell within you want to put.

Hooray! And the code is ... where? We are anxious to see this. :-)

Cheers,

David

--

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

---

---

Subject: Re: Similar to a Widget\_Container  
Posted by [Antonio Santiago](#) on Fri, 27 Feb 2004 14:12:01 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Sorry :)

Well, the code is pretty bad and it's comments are in spanish but from now it does i want.

I attach de code an example 'main' file to see how to use. Depend on size of widgets to put inside it can seem extrangs results but i think is good.

Bye

If you will be rich with it remember your friend Antonio ;) (or if you need a programmer)

David Fanning wrote:

> Antonio Santiago writes:  
>  
>  
>> Hi, I just finish an object 'boxtable'.  
>>

>> With it you can create a widget\_base and an imaginri table and then you  
>> can put another widget only specifying de cell within you want to put.  
>  
>  
> Hooray! And the code is ... where? We are anxious to  
> see this. :-)  
>  
> Cheers,  
>  
> David

```
; Nombre: boxtable__define.pro  
; Autor: Antonio Santiago
```

```
;BoxTable permite definir un widget_base con una tabla "imaginaria".  
;El widget queda dividido en filas y columnas, de forma que se pueden  
;agregar otros widgets indicando la celda en la que se colocara.
```

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

```
;Inicializacion del objeto  
function BoxTable::Init, parent, n_columns, n_rows, type, _EXTRA=e
```

```
;Necesario indicar el widget padre.  
if(KEYWORD_SET(parent) EQ 0) then return, 0
```

```
if(KEYWORD_SET(n_rows) EQ 0) then n_rows = 1  
if(KEYWORD_SET(n_columns) EQ 0) then n_columns = 1  
if(KEYWORD_SET(max_children) EQ 0) then max_children = 1  
if(KEYWORD_SET(type) EQ 0) then type = 0
```

```
self.parent = parent  
self.boxtable = widget_base(parent, _EXTRA=e)  
self.n_columns = n_columns  
self.n_rows = n_rows  
self.type = type  
self.child = ptr_new( REPLICATE({children}, n_columns, n_rows) )
```

```
return, 1  
end
```

```
;Eliminamos contenido dinamico del HEAP  
pro BoxTable::Cleanup  
ptr_free, self.child  
end
```

```
;Obtine valores de las propiedades.  
pro BoxTable::GetProperty, PARENT=parent, ID=id, N_COLUMNS=n_columns, $  
  N_ROWS=n_rows, TYPE=type
```

```
  parent = self.parent  
  id = self.boxtable  
  n_rows = self.n_rows  
  n_columns = self.n_columns  
  type = self.type  
end
```

```
pro BoxTable::Add, widget, xini, yini, xfin, yfin
```

```
  if( (xini EQ xfin) OR (xini GT xfin) ) then xfin = xini + 1  
  if( (yini EQ yfin) OR (yini GT yfin) ) then yfin = yini + 1
```

```
  geo = widget_info(widget, /GEOMETRY)
```

```
  (*self.child)[xini,yini].id = widget
```

```
;Guardamos ID del widget en esa posicion y el xini y yini al  
;que hacen referencia.
```

```
for y=yini, yfin-1 do begin  
  for x=xini, xfin-1 do begin  
    (*self.child)[x,y].xsize = geo.xsize  
    (*self.child)[x,y].ysize = geo.ysize  
    (*self.child)[x,y].xoffset = geo.xoffset  
    (*self.child)[x,y].yoffset = geo.yoffset  
    (*self.child)[x,y].xini = xini  
    (*self.child)[x,y].yini = yini  
    (*self.child)[x,y].xfin = xfin  
    (*self.child)[x,y].yfin = yfin  
    (*self.child)[x,y].set = widget  
  endfor  
endfor
```

```
if ( self.type EQ 0) then self->MakeHeterogeneo $  
else self->MakeHomogeneo  
end
```

```
;-----  
;Establece los offsets de los widgets para que queden situados segun  
;lo indicado en forma de tabla.
```

pro BoxTable::MakeHomogeneo

```
for y=0, (self.n_rows-1) do begin
  for x=0, (self.n_columns-1) do begin
    base_x = 0
    base_y = 0

    if( (*self.child)[x,y].id NE 0) then begin
      for i=0, (x-1) do base_x = base_x + self->max_xsize_column(i)
      for i=0, (y-1) do base_y = base_y + self->max_ysize_row(i)

      id = (*self.child)[x,y].id
      widget_control, id, xoffset=base_x, yoffset=base_y
;print, 'id='+string(id)+' x='+string(x)+' y='+string(y) + ' base_x='+string(base_x)+'
base_y='+string(base_y)
      endif
    endfor
  endfor
;stop
end
```

;Dada una fila, la recorre de izquierda a derecha, y devuelve  
;el valor maximo de y\_size

function BoxTable::max\_ysize\_row, row

```
max = 0
for i=0, (self.n_columns-1) do begin
  ;Si el valor es 0, es que no hay widget en esa celda. '0' es el
  ;valor de inicializacion.
  if( ((*self.child)[i, row].ysize GT max ) AND $
    ((*self.child)[i, row].id NE 0) ) then $
    max=(*self.child)[i, row].ysize
  endfor
return, max
end
```

;Dada una columna , la recorre de arriba a abajo, y devuelve  
;el valor maximo de x\_size

function BoxTable::max\_xsize\_column, column

```
max = 0
for i=0, (self.n_rows-1) do begin
  ;Si el valor es 0, es que no hay widget en esa celda. '0' es el
  ;valor de inicializacion.
  if( ((*self.child)[column, i].xsize GT max ) AND $
    ((*self.child)[column, i].id NE 0) ) then $
```

```

    max = (*self.child)[column, i].xsize
endfor
return, max
end

```

```

;-----

```

```

;Establece los offsets de los widgets para que queden situados segun
;lo indicado en forma de tabla.
;Permite sitiar juntos widget de una misma fila o columna
pro BoxTable::MakeHeterogeneo

```

```

for y=0, (self.n_rows-1) do begin
  for x=0, (self.n_columns-1) do begin
    if( (*self.child)[x,y].id NE 0) then begin

      id = (*self.child)[x,y].id

      xini = (*self.child)[x,y].xini
      xfin = (*self.child)[x,y].xfin
      yini = (*self.child)[x,y].yini
      yfin = (*self.child)[x,y].yfin
;print, 'xini='+string(xini)+' xfin='+string(xfin)+' yini='+string(yini)+' yfin='+string(yfin)
      base_x = self->max_xsize_offset(xini, xfin, yini, yfin)
      base_y = self->max_y_size_offset(xini, xfin, yini, yfin)

      widget_control, id, xoffset=base_x, yoffset=base_y
;print, 'x='+string(x)+' y='+string(y) + ' base_x='+string(base_x)+' base_y='+string(base_y)
    endif
  endfor
endfor

;stop
end

```

```

;Devuelve el xoffset para la posicion inicial de un widget
function BoxTable::max_xsize_offset, xini, xfin, yini, yfin

```

```

max = 0
for row=yini, yfin-1 do begin
  tam = 0

  ;por encima.
  for colum=0, xini-1 do begin
    ;Obtenemos ID, SET actuales y el SET de la columna anterior.
    act_id = (*self.child)[colum, row].id
    act_set = (*self.child)[colum, row].set

```

```

if( colum EQ 0 ) then begin
  prev_id = 0
  prev_set = 0
endif else begin
  prev_id = (*self.child)[colum-1, row].id
  prev_set = (*self.child)[colum-1, row].set
endelse

```

;Comparamos con la celda anterior para saber si tenemos que

```

if( act_id EQ act_set ) then $
  tam = tam + (*self.child)[colum, row].xsize

```

```

if( (act_id NE act_set) AND (act_id NE prev_id) AND (act_set EQ prev_set) ) then $

```

```

if( (act_id NE act_set) AND (act_id EQ prev_id) AND (act_set EQ prev_set) ) then $

```

```

if( (act_id NE act_set) AND (act_set NE prev_set) ) then $
  tam = tam + (*self.child)[colum, row].xsize
endfor

```

```

if( max LT tam ) then max = tam
endfor

```

```

return, max
end

```

;Devuelve el yoffset para la posicion inicial de un widget  
function BoxTable::max\_y\_size\_offset, xini, xfin, yini, yfin

```

max = 0
for colum=xini, xfin-1 do begin
  tam = 0

```

```

;hay antes del widget.
for row=0, yini-1 do begin
;Obtenemos ID, SET actuales y el SET de la columna anterior.
act_id = (*self.child)[colum, row].id
act_set = (*self.child)[colum, row].set
if( row EQ 0 ) then begin
  prev_id = 0
  prev_set = 0
endif else begin
  prev_id = (*self.child)[colum, row-1].id
  prev_set = (*self.child)[colum, row-1].set
endelse

```

```

;Comparamos con la celda anterior para saber si tenemos que
;print, string(act_id)+string(act_set)+string(prev_id)+string(prev_s et)
;print, 'x='+string(column)+'y='+string(row)
  if( act_id EQ act_set ) then begin
    ;print, 1
    tam = tam + (*self.child)[column, row].ysize
  endif

  if( (act_id NE act_set) AND (act_id NE prev_id) AND (act_set EQ prev_set) ) then begin
    ;print, 2

  endif

  if( (act_id NE act_set) AND (act_id EQ prev_id) AND (act_set EQ prev_set) ) then begin
    ;print, 3

  endif

  if( (act_id NE act_set) AND (act_set NE prev_set) ) then begin
    ;print, 4
    tam = tam + (*self.child)[column, row].ysize
  endif

;print, 'tam='+string(tam)
endfor

;nos quedamos con el maximo.
if( max LT tam ) then max = tam
endfor

return, max
end

```

```

;Definicion de las estructuras del objeto.
;children - Estructura para cada widget hijo
;boxtable - Objeto BoxTable
pro BoxTable__define

```

```

void = { children, $
  id:0L, $
  xsize:0L, $
  ysize:0L, $
  xoffset:0L, $
  yoffset:0L, $
  xini:0L, $

```

```

yini:OL, $
xfin:OL, $
yfin:OL, $
set:0 $
}

void = {boxtable, $
parent:OL, $ ;ID widget padre
boxtable:OL, $ ;ID boxtable widget
n_rows:0, $
n_columns:0, $
type: 0, $ ;0 - Heterogeneo, 1 - Homogeneo
child: ptr_new() $
}
end

```

```

pro init
common share, container
end

```

```

pro click, evt
common share

```

```

print, 'click', evt

```

```

w = widget_info(container, /child)
print, w, container

```

```

cont_geo = widget_info(container, /geometry)
win_geo = widget_info(w, /geometry)

```

```

print, cont_geo, win_geo

```

```

widget_control, w, xoffset=(win_geo.xoffset+10), yoffset=(win_geo.yoffset+10)
;widget_control, container, /realize

```

```

end

```

```

pro main
common share

```

```

prog = widget_base(title='hola', xsize=400, ysize=400, mbar=menu, ROW=0)

```

```
boton = widget_button(menu, value='hola')
boton = widget_button(boton, value='adios', event_pro='click')
```

```
o=obj_new('boxtable', prog, 4, 5, 0)
o->getproperty, id=i
```

```
widget_control, prog, /realize
```

```
win = widget_base(i, xsize=50, ysize=75, frame=1)
o->add, win, 0,0,1,2
;stop
```

```
win = widget_base(i, xsize=150, ysize=150, frame=1)
o->add, win, 1,0,3,2
;stop
```

```
win = widget_base(i, xsize=150, ysize=150, frame=1)
o->add, win, 0,2,2,4
;stop
```

```
win = widget_base(i, xsize=55, ysize=65, frame=1)
o->add, win, 3,0,4,1
;stop
```

```
win = widget_base(i, xsize=75, ysize=50, frame=1)
o->add, win, 2,2,3,3
;stop
```

```
win = widget_base(i, xsize=75, ysize=50, frame=1)
o->add, win, 3,2,4,4
;stop
```

```
xmanager, 'main', prog
```

```
end
```

## File Attachments

---

- 1) [boxtable\\_\\_define.pro](#), downloaded 101 times
  - 2) [main.pro](#), downloaded 91 times
- 

---

Subject: Re: Similar to a Widget\_Container

---

Posted by [David Fanning](#) on Fri, 27 Feb 2004 14:44:57 GMT

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

---

Antonio Santiago writes:

- > Well, the code is pretty bad and it's comments are in spanish but from
- > now it does i want.
- >
- > I attach de code an example 'main' file to see how to use. Depend on
- > size of widgets to put inside it can seem extrangs results but i think
- > is good.

Chairman Emeritus Coyote asked me to pass this along.

"While I was greatly impressed with the use of an object in a purported widget program, the presence of a common block in the main code has ruled this submission out as a finalist in the 2004 IEPA code competition."

- > If you will be rich with it remember your friend Antonio ;) (or if you
- > need a programmer)

The Chairman says he'll get back to you soon! :-)

Cheers,

David

--

David Fanning, Ph.D.

Fanning Software Consulting

Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

---