Subject: Re: widgets and objects
Posted by mirko_vukovic on Fri, 14 Aug 1998 07:00:00 GMT
View Forum Message <> Reply to Message

In article <35D32FC5.568C67E5@astro.uni-bonn.de>,
  Reinhold Schaaf <schaaf@astro.uni-bonn.de> wrote:

!!!!!NOTE

I (Mirko Vukovic) recieved the following e-mail from Reinhold with the request to post it as he had trouble reaching his news-server. I see his message on the group, but in an intelligable font. Thus, I am reposting his email to me.

And now, heeere's REINHOLD!!

I worked quite a bit in this sort of programming and came up with the following scheme, which
was initialized by a remark of Mark Rivers from the University of Chicago:

The first part of the scheme is to put a reference of the object into the widget's UVALUE. The
following part of the implementation of the class CWidget (which is used as an abstract base
class for all sorts of other widget classes) shows how this can be done during initialization.
Whenever the widget is needed, the method CWidget::GetBase returns it, also for all classes
derived from CWidget (unless you override GetBase, what you should'nt).

```
    PRO CWidget__Define
      struct =  { CWidget,         $
                  wBase:0L,        $
                  oFrame:OBJ_NEW()  $ ; The object of class CFrame that
holds the TLBase
                }
    END

    FUNCTION CWidget::Init, oParent, $           ; either CWidget or
CFrame object
                        FRAME=bFrame, $       ; set for frame
around widget
                        XOFFSET=fXOffset, $   ; in pixels relative
to parent
```

```
                    XSIZE=fXSize, $       ; in pixels
                    YOFFSET=fYOffset, $   ; in pixels relative
to parent
                    YSIZE=fYSize          ; in pixels

    ;some checks deleted

      wParent = oParent->GetBase()
      self.wBase = WIDGET_BASE(wParent, $
                        FRAME=bFrame, $
                        XOFFSET=fXOffset, $
                        XSIZE=fXSize, $
                        YOFFSET=fYOffset, $
                        YSIZE=fYSize)
      WIDGET_CONTROL, self.wBase, SET_UVALUE=self
      self.oFrame = oParent->GetFrame()                    ; <= it
happens here
      RETURN, 1
    END

    FUNCTION CWidget::GetBase
      RETURN, self.wBase
    END
```

The second part is event handling: This is all done in a central event-handling routine, named
CFrame_Event, which must be declared as the event handler in all XMANAGER calls. This is of
course a global function, but its only purpose is to distribute the events to member functions
of CFrame:

```
    PRO CFrame_Event, sEvent

      stEventName = TAG_NAMES(sEvent, /STRUCTURE_NAME)
      oFrame = CFrame_GetFrame(sEvent.top)

      bEventHandled = 0B

      CASE stEventName OF

        ; Events from base widgets
        'WIDGET_BASE': $
          BEGIN
            bEventHandled = oFrame->OnResize(sEvent.id, sEvent.x,
sEvent.y)
```

```
        oFrame->OnUpdate
      END
    'WIDGET_KBRD_FOCUS': $
      CASE sEvent.enter OF
        0: bEventHandled = oFrame->OnLooseKbrdFocus(sEvent.id)
        1: bEventHandled = oFrame->OnGainKbrdFocus(sEvent.id)
        ELSE:
      ENDCASE
    'WIDGET_KILL_REQUEST': $
      bEventHandled = oFrame->OnKillRequest(sEvent.id)

    ;Events from button widgets
    'WIDGET_BUTTON': $
      CASE sEvent.select OF
        ; The frame handles all events, could be solved differently

        0: bEventHandled = oFrame->OnButtonRelease(sEvent.id)
        1: bEventHandled = oFrame->OnButtonPress(sEvent.id)
        ELSE:
      ENDCASE

  etc etc
```

The class CFrame, which is used as an abstract base class for other classes, which implement
concrete frames, provides only non-functional event-handlers:

```
  FUNCTION CFrame::OnResize, wID, iX, iY
    RETURN, 0B
  END

  FUNCTION CFrame::OnButtonPress, wID
    bEventHandled = 0B
  END

  etc.
```

The real work is done in the OnResize function of the concrete frame:

```
  FUNCTION CMainFrame::OnResize, wID, iX, iY

    ;the work is done here:

    lXSizeReq = LONG(iX - self.oDraw->GetXOffset() - self->GetXPad())
```

```
    lYSizeReq = LONG(iY - self.oDraw->GetYOffset() - self->GetYPad())

    lXSize = self.lXSizeMin > lXSizeReq
    lYSize = self.lYSizeMin > lYSizeReq

    self.oDraw->SetXSize, lXSize
    self.oDraw->SetYSize, lYSize

    RETURN, 1B
  END

  FUNCTION CMainFrame::OnButtonPress, wID

    WIDGET_CONTROL, wID, GET_UVALUE=stButtID ; the ID of the button
pressed
    CASE stButtID OF
      'wButtSetMen': $
        BEGIN
          IF NOT self->bModeIsSet() THEN BEGIN
            oData = self->GetData()

    etc.
```

This scheme has the advantage that all routine work with event handling is concentrated in the
Class CFrame. If one implements a concrete frame class, one has only to provide the functions
CMyFrame::OnResize etc with the functionality, which is needed for this concrete class. One
can forget all the trouble in CFrame, once it is implemented prooperly!

I hope that my answer gives you some idea how one could procede. I have developed with this
(and some other) concepts the first stage of a quite complex user interface. But still a lot
of work (e.g. with positioning of widgets) has to be invested into this framework. And so (sad
enough, but that's life), since IDL 5.1 supports ActiveX, I will almost certainly abbandon
widget-object programming in IDL and switch to VC++ + IDL as ActiveX server.

Best regards

Reinhold

--

****
Reinhold Schaaf
Ettighofferstr. 22
53123 Bonn
Germany

Tel.: (49)-228-625713
Email: schaaf@astro.uni-bonn.de
****