Subject: Re: Trouble with IDL 5.0.2 (filled contours) unsolved?
Posted by Reinhold Schaaf on Wed, 04 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> By the way, I've been writing object-oriented programs
> this week and I am absolutely blown away by how powerful,
> flexible, and simple these programs can be. This is the best
> thing RSI has EVER added to IDL with the exception of widgets.
> If you haven't yet tried to write an OOP program, give it
> a go. I guarantee you will be impressed with it.
>
> (I'm not talking about the object graphics class library,
> which is much more difficult to learn. But just writing
> programs as objects with their own methods, etc. I spent the
> last two days writing a plot object that uses direct graphics
> calls. It does things that I didn't think were possible
> in IDL! And every time I make it do something else, I get
> about 10 new ideas for something ELSE it can do. I like it.)
>

As a newcomer to IDL with a background in C++, I more or less share David's opinion on object oriented programming in IDL. I am programming a (simple) user-inteface for data-reduction software, tried object graphics, found it full of bugs (or unexpected features?), and settled on a combination of direct graphics and object oriented programming.

When I started digging into IDL, I could not believe that object oriented programming was possible without any typechecking. Well, it can be done (of course one must be prepared to do a lot of testing and debugging which would be unnecessary in C++). As a second point, one really has to get used to the fact that objects cannot be created on the stack, (stack objects are destructed automatically on exit from the program unit in which the object was constructed). Hence one is forced to destruct any object manually, which is highly error-prone and makes life really uncomfortable.

A third remark concernes event handling: It is not possible to define a method of a class as the event handler of a widget. Consider:

```
PRO CFrame__Define
  struct =  { CFrame,     $
            wBase:0L,   $
            wDraw:0L    $
          }
END
```

```
FUNCTION CFrame::Init
  self.wBase = WIDGET_BASE()
  self.wDraw = WIDGET_DRAW(self.wBase)
  WIDGET_CONTROL, self.wBase, EVENT_PRO='CFrame::Event'; Not allowed!
  WIDGET_CONTROL, self.wBase, /REALIZE
  RETURN, 1
END

PRO CFrame::Event, sEvent
  ;handle events
END
```

This leads to the runtime error:

```
% WIDGET_CONTROL: Object method is not allowed in this context.
```

As a consequence, one is forced to make the event handler a global function. But global functions cannot access members of objects, so one has to add methods to the class which would be unnecessary otherwise. I.e. handling resize events in the example requires:

- A way to get the CFrame object that contains the widget sEvent.id.
  Therefore one has to maintain a list of all existing CFrame objects.
  This list has to be supplied with methods to search for a widget in its
  members.

- CFrame::Resize, xSize, ySize, in which the actual resizing is done.

Things could be ways simpler if global functions were allowed to be event handling routines! I wonder whether this could be changed in future versions of IDL.

Bye

   Reinhold

--

 ** *
 Reinhold Schaaf
 Ettighofferstr. 22
 53123 Bonn
 Germany

 Tel.: 0049-228-625713
 Email: schaaf@astro.uni-bonn.de
 ** *