Subject: Re: Common or not common
Posted by mgs on Wed, 19 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

In article <879774434.18173@dejanews.com>, meinel@aero.org wrote:

> davidf@dfanning.com (David Fanning) writes:
>>
>> Mr. Morisset writes:
>>
>>> I can't imagine passing 25 or more parameters to the routines.
>>
>> Nor can I. Although event handlers often need at least this
>> much information to function properly. You can pass the
>> information by common blocks, which has many, many limitations
>> or you can pass the information via "pointers". I prefer the
>> latter.
>>
>
> Talk to C programmers. Pointers are a two-edged sword. You can do
> pretty neat stuff with pointers, but they are also the source of
> about half of the bugs.

No doubt about it. However the amount of code they save reduce the number
of possible bugs by at least half. It's a matter of debugging a few lines
of not terribly intuitive code vs. debugging dozens of lines of only
slightly more understandable code.

...
> I currently use commons and named structures. There are times
> when I don't know the size of some arrays until well after the
> structure is created. However, I just read in the manual (TFM)
> that anonymous structures _can_ be modified after creation.
> I'll have to give that a try...

For either structure you can use a pointer to your data.

> Which brings up an IDL gripe -- why do named structures have
> different properties than anonymous structures?

They each have their advantages. I could usually care less about the names
of my structures, yet I depend on the named event structures to determine
the type of events being handled by the user interface. Here's a portion
of my user interface event handlers:

```
PRO IasGps_Event, event
   sEventType = Tag_Names(event, /Structure_Name)
   CASE sEventType OF
      'WIDGET_KILL_REQUEST': BEGIN
```

```
        ;   code
        END
      'WIDGET_TRACKING': BEGIN
        ;   code
        END
      ELSE: BEGIN
        ;   code
        END
  ENDCASE
END
```

The idea I use is that IDL's named event structures will allow different types of events to be processed. Without them, I'd have to parse out elements of the structure to see what was actually intended. By using the ELSE I can pass in my own unnamed structures as events, since I know they'll be picked up in the ELSE section.

> And why aren't
> structure properties consistent between types? According to TFM:
>
> "Once defined, a given named structure type cannot be changed."
>
> Yet in the given example, NAME is defined as a zero-length string,
> but subsequent assignments can have any number of characters. Are
> strings the only variables that can be changed in a named structure?
> Why are strings handled differently? Should I make everything in my
> named structure a string and then convert to some other type at the
> appropriate time? Inquiring minds want to know.

Strings are treated different. Consider it as a pointer to a string, without having to worry about the pointer. Very much like the difference between char and string. Define the structure as it is needed, don't use all strings. That would be highly inefficient, especially for your image data.

>> You miss the ability to have more than one version of your
>> program running at any one time. How good is that great image
>> processing routine if you can only process one image at a time?
>
> That's odd. I have a "great image processing routine" that uses
> commons and I can process more than one image at a time. Maybe
> I'm just "smarter than your average bear" (for those of you who
> didn't watch Hanna-Barberra cartoons, that was a quote from Yogi
> Bear).

Actually, I think you may have misinterpreted David's question. If I start multiple versions of an application from the command line (since the command line can now be active with the XManager changes), they do not

interfere with each other. They are two entirely different applications. This is not possible with commons since each instance of the appication would be using the same common block. Consider compound widgets for a moment. If you use something like the CW_BGroup in several areas of your application, does selecting one button ina button group directly affect the data of the other button groups? The answer is no, because they are not using common blocks.

Back to your example, what happens when you apply a smoothing function to one image and an edge detection to the other? If they are the same data in the common block you'll get an edge detection of your smoothed data. You can get around this with pixmaps and copies of the original data, but you're taking extra steps to avoid something which can be done more appropriately without commons.

--

Mike Schienle Interactive Visuals
mgs@sd.cybernex.net http://ww2.sd.cybernex.net/~mgs/