Subject: Re: IDL Style (was Re: IDL 5.2 GUI Builder Tutorial ?)
Posted by mgs on Thu, 27 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <7ik7fu$60q@post.gsfc.nasa.gov>, fireman@mcst.gsfc.nasa.gov (Gwyn Fireman) wrote:

> Mike Schienle (mgs@ivsoftware.com) wrote:
> : dealing with a very disgusting usage of WIDED. Our tax dollars (in the
> : U.S.) have been used to create this:
> : base1 = WIDGET_BASE(base0, /ROW)
> : label1 = WIDGET_LABEL(base1, VALUE = 'Input L0 file ')
...
>
> Mike -
>
> What exactly is so bad about this?

Hi Gwyn -

No clue what base0, base1, label1, etc. indicate, and they're repeated throughout the example, and throughout dozens of files in the project. Something descriptive is really useful, especially when you're dealing with a lot of files, all of which look identical, and all of which have no reasonable amount of comments. Literally, there is about a 100:1 ratio for code:comments in the project. It's the kind of code slapped together in an afternoon with no consideration for the people to follow along afterwards and make changes, enhancements, deletions, etc. How would you like to have to maintain several thousand lines of that code for six months or five years? The amount of time it took me to chase down nondescriptive, uncommented code was very wasteful and unproductive. Of course, there are dozens of people just like myself having to wade through the same plate of spaghetti. If it had been done correctly at the start, the costs and time could have been greatly minimized for myself and the others. It reminds me of the old FRAM filter commercials, "Pay me now, or pay me later." The up-front costs are minimal compared to the costs for changes at a later time.

> Aside from the fact that the input
> windows don't line up. It looks like my widget code - well, the last
> I wrote anyhow, which was well before GUI Builder, before WIDED, and even
> before compound widgets.
>
> : ...let me suggest an IDL Style Guide that I put together earlier this year.
> : <http://www.ivsoftware.com/IDL_Style.html>.
>
> This style guide is enforceable within an organization, but there is
> too much variation in personal style for the IDL community to adopt in

> en masse.

I readily admit it's a huge pain in the ass to code as presented in the style guide. But, consider the situation where you are asked to come in and make changes to code that has followed a style guide like I presented. Would you prefer to chase down page after page of base0, base1, label1, or wBaseMain, wBaseStats, wLabelView? That's just the tip of the iceberg, of course. If you are a manager, would you rather be charged $100 per hour (staff or consulting costs are very similar to managers when you consider the different overhead rates), to develop code that can be easily navigated and modified for the next release, or would you prefer to have to pay to redevelop large portions because the new programmers can't figure out what the original programmers had in mind. It's a sure way to have dead code strewn throughout your programs because no documentation points out where the pieces go, or what they are intended to accomplish. It's the difference between coding for instant gratification, vs. coding for long-term development.

Let's take this one more step. A step which I have not seen work yet. You might recognize it as a Matrix organization.

All of our personal styles are very different. I prefer my own style, just as you prefer yours. From a long-term view, or from the view of someone holding the purse-strings, this has no benefit. Suppose you had a group of people, working on different projects. Gwyn is (hypothetically) an expert at manipulating data streams at the bit level, David is a GUI specialist, and Bill is a wizard with math functions. If each of these people followed the same standards and could quickly comprehend the reasoning behind the other's code, they could be used to their best interest or capacity. Plug and play engineers, if you like. As a single programmer, I don't relish that thought. As a programmer working within this group, I would love to be able to look at any function or piece of code and quickly recognize its purpose, almost as if I had written it. Again, I haven't seen this in practice, but I've been in a couple groups attempting to make it work.

> For example, I reject the idea that variable names should
> start with a letter corresponding to data type.

Keep in mind, I'm not pushing to have everyone adopt this style guide as is, I'm looking for comments to improve it to the point where a large group could use it. I appreciate your comments about variable names. Considering IDL makes it so easy to modify data types, I think it's beneficial to imply the data types in the variable names. That's not a drawback to IDL, just a fact that we have to keep in mind when we use it. I'm very happy that I can change an IDL data type in one line, whereas ADA would take a full page of code to accomplish the same thing. The downside is the number of times I've tried to do something with a scalar, but IDL had converted it to an array, or similarly when my ints have been

converted to floats. These little things can smack you when you least
expect it.

> -- General Sciences Corporation / MODIS Characterization Support Team

As a coincidence to Gwyn's project at GSC, this Style Guide was originally
developed for the MODIS algorithm testbed while I was working at Hughes
SBRS. It's very loosely based on the Hughes LOI (can't remember what LOI
stands for) for C programming. The LOI was developed to provide ISO 9000
approval.

--
Mike Schienle Interactive Visuals, Inc.
mgs@ivsoftware.com Remote Sensing and Image Processing
http://www.ivsoftware.com/ Analysis and Application Development

---

Subject: Re: IDL Style (was Re: IDL 5.2 GUI Builder Tutorial ?)
Posted by Michael Asten on Tue, 01 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

>
>>
>> Mike -
>>
>> What exactly is so bad about this?
>
> Hi Gwyn -
>
> No clue what base0, base1, label1, etc. indicate, and they're repeated
> throughout the example, and throughout dozens of files in the project.
> Something descriptive is really useful, especially when you're dealing
> with a lot of files, all of which look identical, and all of which have no
> reasonable amount of comments. Literally, there is about a 100:1 ratio for
> code:comments in the project. It's the kind of code slapped together in an
> afternoon with no consideration for the people to follow along afterwards
> and make changes, enhancements, deletions, etc.

Hi guys,
arent we confusing a couple of different issues here?

Firstly, thanks to Mike S. for the earlier posting on the style guide. I like
it. I tend to use a default fortran-style naming convention, but the additional
rigor suggested by Mike makes sense. In hindsight it would have saved me a few
"what did I mean when I wrote this 3 months ago" experiences.

Secondly, regarding undocumented widget code generated by WIDED or GUIBUILDER, I
don't accept the argument that poor documentation is an argument

against their use. Their use saves a great deal of programming time for some of us, and before we give away, (or archive for ourselves!) the code, it is commonsense professsional practice to add blocks of comment code. This will be "what the widget does" rather than individual lines of "how the code does it". Mikes angst with the busy person who failed to document before putting code into the public domain, is justifiable, but keep in mind that a lot more tax dollars might have been spent if the programmer had worked from a lower level.

I believe that arguments against the use of guibuilders are akin to arguments against the use of 4GLs generally (amazing the number of professional programmers who think that programming in any language higher-level than c++ is for wimps).

One of the beauties of machine generated widget code, apart from initial time savings, is that it is predictable and systematic and requires less documentation than a block of code written by a person with a personal style. However one has to resist the temptation to hack the generated code to pieces. I think the guibuilder is a good attempt in the right direction, since it encourages separation of event handling code from gui code. I also find that the much-maligned WIDED is superior when it comes to making a gui to handle text fields (especially real mumbers) . I'd love to see rsi produce a clear tutorial on how to get the most out of the guibuilder, to the same standard as say David Fannings tutorials on how to avoid it.

Regards,
Michael Asten
