Subject: Re: Tip: how to mix object gui with command line
Posted by Terry Smith on Thu, 23 Dec 1999 08:00:00 GMT

In article <MPG.12c81f5e6aecc9239899c7@news.frii.com>, davidf@dfanning.com
(David Fanning) wrote:

```
> Karri Kaksonen (karri.kaksonen@picker.fi) writes:
>
>> I just thought of dropping a line about a discovery I made.
>> This may be old news for all old timers but as it was new for
>> me I thought of sharing this idea.
>>
>> When you build an application as an object like:
>> o=obj_new('MyApplication')
>>
>> and you start up the graphical user interface with buttons etc.
>> o->draw
>>
>> then I suddenly notice that there is something weird going on
>> and want to have a look at my data from the command line.
>>
>> In a widget-program I would have to quit the program and start
>> debugging. But in an object program I can leave the program
>> running and just fetch the data.
>>
>> The key is in coding in methods for accessing private stuff like:
>>
>> function MyApplication::getdata
>>     return *self->data
>> end
>>
>> then I can just click on the IDL> command line and write:
>> a=o->getdata()
>> and continue to work on it on the command line.
>
> I think you meant "RETURN, *self.data".
>
> But in any case, this is certainly possible to do. I would
> be just a bit careful with it, however, since it is quite easy
> to completely defeat the whole purpose of objects, which
> is to encapsulate the data and the methods that work on
> the data inside the object, out of the view of the rest
> of the world.
>
> For example, you can easily write a GetDataPointer method:
```

```
>
>    FUNCTION JUNKER::GetDataPointer
>    RETURN, self.data
>    END
>
```

> Now the outside world can muck around with the data *inside*
> the object. Oh, dear! Keep in mind that just because something
> is *possible* doesn't mean it's always a good idea. :-)
>
> Cheers,
>
> David
>

I think that you've fallen victim to the seductive lure of
hyper-encapsulation; the same lure which kept RSI from implementing public
class data. In most OO languages, it is recognized that certain members of
the class variable set should be available for public consumption. In good
programs, these are usually flags, often simply Boolean, rather than complex
data structures whose modification would more simply/reliably be handled by
a special-purpose method... they are things which are pivotal to the
external functioning of a class, and are not at risk of reworking.

The trivial "GetProperty" and "SetProperty" methods associated with most
classes are an example of what we are forced to do to overcome this
limitation. They often do nothing but return a single class variable:

```
pro foo::GetProperty, X=x
   if arg_present(x) then x=self.x
   return
end
```

So instead of saying simply obj.x (as if it were a structure), I have to say
obj->GetProperty, X=x, which doesn't fit well into compound structures like:

```
res=obj1.X+obj2.Y+obj3.Z
```

Now, of course, enforcing encapsulation will prevent users from abusing
public class variables, defeating the advantages they might have had
otherwise, but shouldn't we be given the choice, at least from an efficiency
point of view.

Just my two cents.

JD