Subject: Re: C++ FQA
Posted by Mindtraveller on Mon, 12 Nov 2007 14:54:07 GMT
View Forum Message <> Reply to Message

It is funny, I`m criticizing the tool, I`ve been using for years. Just to make clear - programms written under BCB were my main job throughout these years. Yes, while I used it heavily, I just wanted something better - and this is purpose of my being here. Event this critics aimed not to blame Borland with VCL, but to think of something better and more advanced than good old VCL.

It was also mentioned above, that VCL (it really stands for Visual Components Library) - is not framework, it is simple bunch of approaches and code for rapid GUI writing. I`ve forgotten this, considering VCL a complete framework - it`s wrong of course.

Quote:I don't see anything bad on loading forms from a resource file and/or from exe file
The main issue is that you need to have all components in your code - because application doesn`t know which components are used at the compile time. This way we have very-big-applicaction-code. Borland solved this by dividing components by packages, which you can plug-in (or not) in your app.
And, yes, they managed to make serialization rather quick. I wonder if we`d have even more effective and fast-starting applications if VCL used more simple ways to start.

The second issue about working with these resources is that heavy graphics usage led to fast-growing exe files - because picture resources where also included into text resources in the form of textual binary data representation:

```
Glyph.Data = {
  76010000424D7601000000000000760000002800000020000000100000000100
  04000000000000010000120B0000120B00001000000000000000000000000000
  800000800000008080008000000080008000808000007F7F7F00BFBFBF000000
  FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00555555555555
  5000555555555555577755555555555550B0555555555555F7F7555555555550
  00B05555555555577757555555555550B3B05555555555F7F557555555555000
  3B055555555557775575555555555500B3B0555555555577555755555555550B3B
  055555FFFF5F7F557555570005000 3B05555577757775575555570BBB00B3B05
  555577555775557555550BBBBBB3B05555557F555555575555550BBBBBBB0555
  55557F55FF557F5555550BB003BB075555557F577F5575F5555577B003BBB055
  555575F7755557F5555550BB33BBB0555555575F555557F555555507BBBB0755
  55555575FFFF7755555555570000755555555557777775555555}
```

The third issue about text resources is that one may "hack" application. Imagine, you have access rights in your banking operations program. Operator access rights don`t allow him to move money from one account to another - where app simply disables unavailable buttons (this is very common approach).
Ok. You just run special resource editor (there are visual ones also) and make button enabled by default. Or add button with the same handlers, or 1000 more ways to change how your program works - without actual debugging, back engineering, etc. Of course, there are 3rd party utilities encrypting .exe, but think of how many programs are written by people who actually can`t imagine that their programms can be hacked in any moment, without even installing a debugger and living

all the application code alone.
Once I was needed to try some application, but it had password protection. I applied some resource hacks and entered it (more just for fun). But I just didn`t like an idea that anyone else may do the same with apps I`ve written.

Quote:Ironically, it seems like U++ starts to be quite ide supported too Yes, but U++ restrictions are way too wider than common IDE`s. Also, I like the way of programming U++ proposes to me. After many problems with standard GUI approaches, to have components in class, where they are needed - is like breath of fresh air for me. That is why I don`t see any problems with U++ restrictions on my code (maybe I`m wrong with this, but alternative restrictions seem to me far worse).
Yesterday I`ve finished rewriting some of my helper classes for U++ (previuosly created for BCB) - the ones I commonly used in my apps. The first version of one of them (ConveyorThread) - being posted into forum. It turned that they were four times smaller, and all my tool application, rewritten under U++, fit into 10Kb of source code (without helper classes). This is more than three times smaller than BCB version - even to mention I`m no professional in U++ for now.

Quote:D does seem good, and its built-in string handling and GC are apparently fast.It was mentioned above that D`s GC may sometimes hang your application for 2-3 seconds. It soesn`t sound like good for serious programms. Also, I don`t like an idea that something else would control freeing of blocks I allocated. I don`t think that good-structured code have much problems with loosing allocated memory. More problematic for me personally is keeping things under control when you heavily use pointers with address arithmetics on many types. It`s very effective and quick (and sometimes necessary) but very dangerous.