Subject: U++ state
Posted by andrei_natanael on Sat, 09 Jan 2010 00:34:30 GMT
View Forum Message <> Reply to Message

Hi,
I'm with U++ since December 13, 2007(bytefield) so I think I've gained a bit of experience with it even if I've only done programs which solve my problems, projects for university and for a non-profit organization. I have some reasons for sticking with U++:

U++ use BSD license
statical linkage
have good examples how to do things
it's *almost complete* cross-platform
have good performance
developers dedicated to work on it
friendly community
some good widgets (ArrayCtrl, GridCtrl)
is quite easy to create new widgets
(almost)have a theming engine
it doesn't impose a single work-flow (i.e. widgets should be allocated only on heap or stack)
it's a project where I've learned a lot of new things
keep increasing usage from new developers

But not everything from U++ is as I wanted so I'll explain "negative" sides which keeps me worry about.

Documentation if exists it's sometime outdated
Dynamic linking to U++ libraries is hard
It only support Windows(+CE), Linux & BSD (IMO 2 platforms if we count X11 as platform instead of Linux & BSD), it should support MacOS to be cross-platform
IMO Chameleon is a good design (ChStyle stuff) but data acquisition for it is a bit of mess because it's not providing the same API for different platform i.e. we have XpImage for Windows, GetGTK for gtk+ **
I find hard(or limited)to create an advanced interface without using layouts(sizers) **
Look and feel is incomplete, for example Scrollbars in Windows Vista and 7 have a special behavior (the buttons from heads are highlighted when mouse is over thumb), U++ implementation of menu for gtk+ is using Windows behavior, if there is not enough vertical space it move a part from menu at a side, U++ doesn't disable Scrollbar head button if the thumb is near it(gtk+) and may I continue with many other aspects. **
It doesn't support receiving events like "theme changed" or "DPI changed" from gtk+/gnome (and partly from Windows) so you have to restart your U++ program in order to use new settings
IMO (probably I'm wrong here) U++ choose bad operator = for PICK, it should do what it say "equality" and that means that what is in one side is in other side too, i would use <<= (deep copy?) operator for PICK so you should not have to invent hacks to avoid picking if you didn't want to use it (is that done to have picking for function returned value?)
Even if macros make our work easier(to acquire RAD) i think there are too much macros in a modern framework as U++ and they hide portions of code making it less readable. I'm pro readability even if that means writing 10 chars or more to get it, let's count some macros:

THISBACK, PTEBACK, INITBLOCK, EXITBLOCK, __countof, NTL_MOVEABLE, FN*, ONCELOCK, INTERLOCKED, CH_STYLE, CH_COLOR, GUI_APP_MAIN, CONSOLE_APP_MAIN and all these macros are from developer space not from U++ core developer space which contain more macros which make core unreadable in some portions (i.e. code responsible for IML files, LAY files, DLI), IMO there are nicer solutions to solve problems.

I know that everyone have limited time and I don't expect any change to come from someone but I'm putting these here to know what to work on in future to have a better U++.

** are stuff which I'm saying that I will work on when have time, but always happen to run out of time

Andrei

P.S.: If you have different view on some stuff feel free to post your opinion. I want to hear what others believe about these.
