Subject: Re: My explaination of why Ultimate++ is not mainstream
Posted by cbpporter on Thu, 14 Aug 2008 19:37:39 GMT
View Forum Message <> Reply to Message

I don't see a failure here either. I'm not aware of any publicity/marketing campaign, or even anything comparable to it, so I can't say we failed. Maybe Mirek is promoting it locally or something.

Sure, we are only a few that use it. Sure, we don't have a Wikipedia entry. Getting more users would definitely be a great. It would help extend U++ and make it even more general and easy to use.

The main advantages of U++ IMO are:

1. It's small, fast, powerful and reasonably crossplatfrom
It is exactly what you need if you want a lean and mean environment in which to work. You get everything out of the box, no need to use makefiles, set up paths, install runtimes.

And even if it is small, it is powerful and fast. It fits well with low processing power and memory requirements. You get to write short code, which does a lot and you have powerful and appropriate widgets. U++ does have some useful stuff for writing console code and containers are better than STL, but GUI code is where U++ strength lies.

With it's lack of popularity, U++ still manages to fill a niece and it was exactly what I needed when I was searching for a new C based toolkit that could replace and keep up with Delphi. IMO, the only other toolkit that can keep up with U++ is Qt. Qt is even better, but it is pretty huge, almost rivaling something which is "managed", like a .NET runtime. Deploying is also harder with Qt, and Windows support was not that great, but I like it in 4.3+. Some might be put of about the licensing cost also.

2. It is smart and clean C++ code
Often people seeing U++ code are surprised how clean it is. For some sections of code, you could swear it is Java code. It is also smart code and uses a lot advanced C++ features. I'm sure it is not even compilable on older and less complete C++ compilers. It also keeps your knowledge fresh. In U++ I found myself writing template code for the first time out of need and because it would simplify stuff, rather than out of necessity to interact with some code. It makes templates fun and useful.

3. It is not under GPL license.
FOSS people might not like this, but this is a huge advantage for me. I'm more pragmatic and don't really care about some people who sat down one day and said: "this is free software". Talking about U++ BSD license, as long as I can take the source code, do absolutely whatever I want to it to deploy my application, this is free software, even though I have to include some copyright notice and acknowledge the source. Of course, I'm not doing anything radical to the source tree, but I do have small patches here and there. This means "free" for me.

It depends a lot on what you think "free" means. I think that GPL is forced freedom, so it is not freedom at all. LGPL on the other hand does give you some extra rights that I need, but I only

applies to dynamic linking, and AFAIK U++ uses static linking.

Anyway, I'm no lawyer, but maybe using BSD license and not BSD like would be a good idea.

And I don't want to start a license flame war .

4. It has a "code in your face" attitude
This can be considered an disadvantage, but I like the fact that the entire source code is a click away. It can be considered a surrogate documentation. And thanks to BLITZ, any modification is soon usable, without having to run makefiles and move libraries around.

But there are also some disadvantages:

1. Documentation is poor
This doesn't affect me that often, because I already found out most of the stuff the hard way, but new users will have problems. I won't say anything more regarding this subject, because we are working on it and I'm sure we'll have good documentation in a not so distant future.

2. It does have it's quirks and idiosyncrasies
Of course, this is just a matter of taste and others might not see it. Also, some parts are not as well designed as others, and some similar components do lack a similar interface, but in the long run it doesn't really affect users IMO. Also, some implementations are overcomplicated for no good reason, while others are just complicated to achieve the necessary functionality. This also contributes to turning new users away. It is just the result of the age of the project and multiple people working on it. Maybe some day we'll have something like 2015.1 Cleaned Up Edition, with deprecated stuff stripped out and all interfaces designed with uniformity in mind.

3. It is C++ code.
I don't have a problem with the language itself, rather with the compilation tool chain. It is extremely unproductive. I'm talking especially about build times. This is such an issue, that I think that there is absolutely no reason to pick up a C++ based library for absolutely no reason. In theory. In practice, you are stuck with all those libraries you need to interact if you don't want to reinvent a huge wheel. So basically you're stuck with C/C++ and must learn to live with it.

There is no other alternative. It will take years until D stabilizes, and maybe Walter Bright will get tired of it until then and return to C++ . And performance issues makes it hard to choose one of the popular and quite pleasant languages, like Ruby (and even then you would need a GUI toolkit; I don't think well have U++ Ruby bindings too soon).

Personally, I would love to see a Free Pascal port of U++. All we need is to gather a large community of Free Pascal enthusiast around U++, teach them C++, and the port will materialize itself in no time .

But seriously speaking, I am satisfied with U++. With better documentation and a better C++ parser coming along it will be all I need for quite some time. So what it it is not that popular. It does pretty much what I and other people that use it want and we'll stick with it! And if it does not do what we want, we can always come to this forum and have a nice chat about it, and if our demands are pertinent (and we have a patch on hand), we can get things changed.