Subject: Re: PR!
Posted by cbpporter on Mon, 23 Mar 2009 09:21:10 GMT
View Forum Message <> Reply to Message

I think we should build on what we have.

1. Logo should remain the same. It's not like we are changing anything in the framework, we are just trying to promote it. But the style of the logo is old. Modern logo's have gradients and realistic shading. Check out what Oxygen/KDE did with their icons, or how Vista updated the icon for Notepad from the Windows 95 look.

I think I saw some updated icons in an old thread somewhere on the forum, but I can't find it again.

Touching up the logo should be no big deal. If one of us can't do it, a struggling young artist will do a great job for a small amount of money.

2. The forum should stay. It is both an adequate platform of communication and an archive of a lot of valuable information. Going through it and archiving some old threads would be in order, but that is a monumental task and I don't think anybody has the resources or patience to do it.

3. We do need some great introductory articles which must address the most important aspects that differentiate U++ from other frameworks, like:
- why do we re implement STL? Couldn't we have developed a compromise solution, using STL as much as possible and adding just a few containers, like Movable enabled ones? Couldn't we have used STL naming conventions and iterators? I know STL containers don't have a virtual destructor, but that doesn't mean you can't inherit and modify behavior. Sure, there are some restrictions, but we could have retrofitted some standard containers with some useful methods.
- why do we use default pick semantics? Won't it cause a lot of problems? Big learning curve and hidden bugs? Design pattern books say that this is a bad practice. And it is in a world of default copy construction, so we need to highlight that the framework is more or less capable of handling pick semantics without issues because if was designed that way. This issue is going to be hard to "market". I've been using U++ for quite some time now, and while I think that copy constructing should not be enforced on a STL level, we should have left = as default copy construct and only use pick with and explicit picking method/operator.
- why don't we have a couple of libs and some .h. Everybody does that. "Source based distribution? Why? I don't want to recompile everything. What is this, the kernel? You're just lazy!". We need to explain the advantages of U++ modularity. Also BLITZ must be handled with care. It is not well received in a business environment and managers tend to avoid it.
- why do we allocate everything on the stack? This makes a lot of common pattern harder to use. U++ doesn't play nice with things like abstract factories if we follow our style, and if we adopt their style, we loose a lot of U++'s advantages and simplicity.
- leaving aside NTL, why do we implement a new set of widgets? We could have used one of the existing toolkits and expand upon it. Even better question: leaving aside details and language differences, what can U++ widgets do that Gtk ones can't? Gtk is used everywhere, and GtkMM should be a better supported equivalent of U++, right? Such questions will arise, and different environments will have different levels of resistance. For Windows development, most people I know that do C++ would ask: "Another set of widgets? What version of MFC is it built upon? I

can't be better than ToolKit Pro."

And the list could go on and on. What I'm trying to say is that U++ is different, and we need to show this and also why it is good thing. If we create false expectations, we won't get anything more that a temporary interest.
