Subject: Good manual/documentation
Posted by Mindtraveller on Mon, 05 Nov 2007 23:37:31 GMT
View Forum Message <> Reply to Message

I think U++ has come to the critical moment when good documentation becoming even more important for it`s users than it`s growing abilities. This thought is crucial to understand.

U++ is great but it is becoming more and more complex with every build. To make first important step to the next level of expansion between people, the most important thing is creation of worthy documentation.

When I say "worthy", I mean the documentation presenting new level of understanding and learning easiness - to deserve being one successful part of new-approach library as U++.

Yes, U++ has some kind of documentation for now. But I think that the way it`s presented and it`s content is highly, critically insufficient for adequate understanding of U++ approaches and abilities by the most of people.

It is a turning-point when library may attract huge amount of programmers and the thing it is needed most is a good documentation. Easy-learning. Detailed. Extensible.

Developers, developers, developers, developers...

OK. I propose slightly different approach than MSDN has, some moments I found adequate, so I`ll compare what I propose to the thing you all know (MSDN I mean).

1. It is great idea to have 3 different ways to access information. By finding appropriate article, by index and by full-text-search.

2. Indexing and search are common, so it`s all obvious here.

3. The main key in making really good docs is a presentation of it`s articles. It is the main (and only - in the beginning) "guiding star", making U++ in users`s eyes a great and easy-in-use library, or making it hell-for-brain. You all understand that most users don`t really like learning library code. Closer we come to "install -> start using instantly" rule, more people we attract.

3.1 It is bad idea to divide subject-close articles into "common" and "knowledge base" as Microsoft did. Moreover, they redoubled search-hell with naming articles like KB73487348971237812378293991283 67 style. Let`s keep in mind that close to problem articles must be easy in reach and easy in search from current article. Example and hint approach must be integrated into article division approach, as ordinary articles, easy in reach from main article.

3.2 I propose dividing U++ functionality into hierarchical list of subjects. The root subjects will be something like:
* Innovative approaches to organize program
* Most common and useful things
* Program control
* Windows and controls

* Charsets and internationalization
* Utility and extension classes
* Compilation and linking
* TheIDE

Human perception rules tell us that having more 7-10 articles in one list is bad idea. User find interface "comfortable" as long as he knows where to click to move one step closer to what he`s searching for. Less user thinks while searching, the better interface is. This means that having 20 articles with good-structured hypelinks is much better than having 2 extremely long and completely unreadable texts. Let`s keep this in mind too.

For example "Most common and useful things" branch will contain things like Containers, Streams, Strings, Serialization, etc.
Strings branch is to contain Strings overview, encoding issues, formatting values, etc. I can propose my own hierarchy, but core authors are to be the ones to make decisions about what to include and where.

3.3 Great. We have hierarchy of subjects. It is time to discuss how they are to be presented. MSDN gives us plain solution - simple tree control, situating in a left corner of the window. This is ridiculous decision because of one simple thing that Microsoft didn`t undestand. When we talk about huge and complex library with vast documentation, easiness of finding article is as important as article itself!
Long-long uniform strings list is NOT the thing user needs to find article easily!

Yes, it`s bad it`s ridiculous, but how to do it better? It`s easy. We already have solution, widely spread, approved by many people through years. I`m talking about classical decomposition of information we find in a site with good design. It means:
1) Consider our articles as huge site with hyperlinks.
2) Include corresponding hypelinks right into the articles. Hierarchy will be presented as number of links from current article.
3) Have navigation string on the top of the document. You all know it`s something like "Main -> Sublevel1 -> Sublevel2 -> Current topic".
This is crucial detail for user to know where he is in any moment, also having ability to come one or more steps back.
4) Examples, hints and issues should be presented as links being natural part of more general article on current subject.

That`s all for general rule. Together with good structure of hierarchy it must make library much closer to users, attracting much more people.

4. The last thing I`d like to say is making docs extensible. I mean it must have mechanizm for users to propose their articles for docs (through website central, for example).
Due to the dynamics of U++ development it is a good idea to have manual extensible in online. It would also be ideally to create synchronization solution for TheIDE to download changed articles into local copy from site.

That is what I think about documentation and U++. It would be great to discuss these things,

because they`re in great importance, really.