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.

Subject: Re: Good manual/documentation
Posted by unodgs on Tue, 06 Nov 2007 14:11:32 GMT
View Forum Message <> Reply to Message

Hi!
Very good sugesstions. I have in my plans to refactor help window a lot (if Mirek won't do it first but I guess there are many things more important to do). And my concept is very similar to yours. So we'll return to this topic when I implement first version of the new help.

Subject: Re: Good manual/documentation
Posted by mirek on Tue, 06 Nov 2007 20:12:02 GMT
View Forum Message <> Reply to Message

Thanks Mindtraveller. You are absolutely right.

This is the current plan:

We have to release U++ soon, if only because too much has changed and was fixed since 2007.1 that it is simply unfair to keep that version as "major". Not many things are needed to make this release now, so I hope we will be fast enought to make it 2007.2...

The main focus of the next release is everything associated with documentation and code navigation, including the documentation itself...

Mirek

Subject: Re: Good manual/documentation
Posted by Mindtraveller on Tue, 06 Nov 2007 21:43:26 GMT
View Forum Message <> Reply to Message

If you need contribution into new help code / subjects structure - I can be in help. Just don`t have very much time for huge amounts of code, but I`d like to be in help with docs as much as I can.

For example, I may think of articles structure and propose you one. And you`ll have some starting point... So, if you need contribution on any stage of making help system - let`s dicuss it.

I just want great framework to have great help system. As innovative and advanced as U++ is.

Subject: Re: Good manual/documentation
Posted by mirek on Wed, 07 Nov 2007 13:38:27 GMT

View Forum Message <> Reply to Message

Mindtraveller wrote on Tue, 06 November 2007 16:43If you need contribution into new help code / subjects structure - I can be in help. Just don`t have very much time for huge amounts of code, but I`d like to be in help with docs as much as I can.

For example, I may think of articles structure and propose you one. And you`ll have some starting point... So, if you need contribution on any stage of making help system - let`s dicuss it.

I just want great framework to have great help system. As innovative and advanced as U++ is.

Well, the only thing to consider is that U++ documentation is bound with packages. That IMO pretty much creates the basic structure.

(We can debate it, but as modular principle, I consider that a good idea, even if sometimes it is a problem if you want to go "wider")

Mirek

Subject: Re: Good manual/documentation
Posted by Mindtraveller on Thu, 08 Nov 2007 07:57:45 GMT
View Forum Message <> Reply to Message

luzrWell, the only thing to consider is that U++ documentation is bound with packages. That IMO pretty much creates the basic structure.

(We can debate it, but as modular principle, I consider that a good idea, even if sometimes it is a problem if you want to go "wider")

Mirek
I`ve looked what structure we`ll have this way:

Examples
Reference
Tutorial
UppSrc

There, Reference will have sub-levels with string-sorted list of widgets and their examples. This maybe makes good Index structure, but in my opinion it is not a good structure for articles at all. Why?

Let`s imagine you`re novice driver, you buy a car and you want to learn driving. You open manual, and what structure you see:
* Accelerometer
* Accumulator
* Air conditioner
* Breaks

* Driving wheel
* Engine
* Glasses
* Ignition
* Indicating lights
* Injector
* Speedometer
* Transmission
* Wheels
* ... <long string-sorted list>

It makes a kind of good reference catalogue, but it has nothing with actually telling driver what is really important to know and how to use it.

Good manual starts from user, not from internal structure of the product. Because less you need to know about product internal structure to use it, the better user interface it has.

The same thought is for U++ (and everything else). "What`s inside of this thing?" - is far not important question for user. The one and only main question user asks is: "How to do what I need with your product?". And this question should be starting point to think of manual structure. Again, articles structure must have nothing with internal U++ structure, it all comes from user needs.

So we imagine ourselves U++ novice programmers who installed it because they want to implement something. And we think, what user need to know first to start using U++ immediately. This is the first root subject. We tell most important and most innovative things to tell user what he really wants to know first and to attract him.

OK, user knows some commons. But it is not really what he wants. He just came two steps closer to what he need. So our task is to make his survey (while answering the Main and Only question "How do I...?") as much obvious and comfortable as possible.

Thinking this way, you come to the descending specification structure. I mean, dividing by problematics and descending sublevels reveal more specifical parts of root subject. I believe this is the best choice to give user answer how he`d do what he needs.

Long lists is the second issue of bad-designed docs. User find logh lists very incomfortable due to the specifics of human perception. User finds much more comfortable to click one link from short list (coming to more specifical and again short links list) than finding something from long list. This is also an internal confidence problem. As long as you know where to go (or to click) - it is comfortable, it is good interface. When you don`t know where to click it is subconciously incomfortable and you get famous "supermarket problem" where customer doesn`t know what to buy.

That is why in my opinion manual structure must have nothing to do with internal U++ packages structure, instead it all constructed from user-specific needs.

Subject: Re: Good manual/documentation

Posted by mirek on Fri, 09 Nov 2007 12:16:44 GMT
View Forum Message <> Reply to Message

Well, the real idea about dividing topics per packages is to provide a documenation on modular basis. E.g. 3rd party packages can contain documentation that immediately is visible in theide help as soon as package is used.

Anyway, I also do agree with your points. IMO, the solution is to have some "metadocumentation" that will organize existing topics, loaded from packages...

Mirek

Subject: Re: Good manual/documentation
Posted by Mindtraveller on Sat, 10 Nov 2007 00:52:26 GMT
View Forum Message <> Reply to Message

luzrWell, the real idea about dividing topics per packages is to provide a documenation on modular basis.

I think this is good idea - making docs modular. I also believe that modular approach is fully appliable to specifics-descending docs I proposed earlier.
So my idea is to integrate these two approaches and I`ll try to propose way of doing it below.

General idea of my approach is that adding new package leads to assembling of package-supplied articles & structure to the existing articles & structure. This operation has trivial algorithm:

```
for (each_article_of_package)
{
   if (article_already_presents)
      Append_article_text_to_the_existing_article();
   else
      Copy_article();
}
```

Maybe it would also be very interesting approach to mark package-blocks with comments, like %_XXXXX_START_% ..... %_XXXXX_END_%

So if we uninstall package or upgrade it, TheIDE simply removes or updates corresponding blocks and we have updated docs for our latest version of package!

It is discussable of course, but I just like the idea of making docs true user-friendly. And keeping them user-oriented at any cost, because it is matter of making U++ one of leading frameworks among others.