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.