Subject: Re: (Possible) Serialization issue
Posted by Mindtraveller on Sat, 01 Nov 2008 10:53:38 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 04 August 2008 01:21Mindtraveller wrote on Sun, 03 August 2008 16:03Thanx. And what about renaming? I understand compitibility issues, but don`t you consider "no surprises" idea importance?Well, I do not see any surprise here. Well, I returned to old topic because could finally summarize what is really a surprise here. The main surprise is when Serialize() is called and you can`t determine if it is REALLY a loading process or a storing. Why is it a problem? Loading (and saving) process may be followed by lengthy GUI updates or some other initializations including file/database work.

For example, on saving, my app generates a number of directories and files, then archives them with external utility. This was thought to be called rarely because it takes moderate amount of time and system resources (and these files have to be kept along a number of app runs). But each time I start application these files are re-created and re-archived because LoadFromFile() calls Serialize() with stream.IsStoring() == true. This situation will be complete surprise for those who use LoadFromFile() and didn`t read this topic.

IMO the idea of U++ error correction on serialize is wonderful. But user usually needs to know a little more about current serialization stage.
OK, I consider strong influence of compatibility requirements, so renaming LoadFromFile() is no more an option. The solution here could be adding some additional function like stream.IsXXXXXXX() which will guide real internal serializarion direction to be used internally by U++, while IsStoring()/IsLoading() could return general "user-level" serialization direction (IsLoading() always == true on LoadFromFile() ).