Subject: Re: Sharing and Locking
Posted by mirek on Sun, 14 Mar 2010 17:58:09 GMT
View Forum Message <> Reply to Message

gridem wrote on Sun, 14 March 2010 08:37The above approach has objects on the heap instead of stack but it has predictable object lifetime. I think that it's the reasonable overhead to solve the considered race condition in case of object destroying.

Mirek, what do you think?

I am still not quite sure what you are trying to solve:)

What I think you are trying to do is to avoid dangling pointer. Anyway, making pointer itself dangling helps only a bit and perhaps is not a good strategy: Pointer itself can still exist, but the state of object can be "destroyed". So it may seal some references to it, but IMO is not a good way.

Now maybe my experiences are not wide enough, but I belive that so far, I had little problems with race conditions of this kind in MT code. I guess, usually the best is to make things simple and not get involved into any shared ownership, which after all is the cornerstone of U++ design.