Subject: Re: Overriding Display methods too complicated due to high amount of arguments
Posted by Klugier on Sat, 14 Nov 2020 19:29:18 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I am happy we are on the same page. I didn't know that this additional parameters were added in the past. Anyway the deprecation mechanism was introduced int c++ since c++14. So, even if the API mistakes were done in the past it can be reversed. The common strategy is to do not remove API immediately. Instead of that mark old method as deprecated and in the message specify that it is deprecated and it will be remove after for example three releases (Optimally it should be specify like 2022.1). This will give people time to migration and always generates warning, so they will know that they need to fix something. Without this one created API would not be able to make a change.

This is common strategy in software world to deal with old API decision that do not fit to current times. Let's just take our favorite language c++ it removes auto_ptr in c++17. It means all code that uses this kind of pointer will not compiler with c++17 standard. The migration is relatively simply and they warn that it will happen in c++14. The same is true for example for Python, they remove unnecessary API within 3 major releases.

Quote:You know, I feel like you are the only one complaining about this API decision and it even looks like the only reason you do is that you have read some well meant rule somewhere and not even read it in details.
I think I follow this rule for some time and I am quite sure that the outcomes are good. In context of software maintainability and easy to use. However to prove legitimacy, a study would have to take place. I think it will not happen soon, but if you ask people how many parameters they want to use they will always answer that less is better.

Quote:What is said in the first link sounds very reasonable to me. Have you read it all?
Yes, I have read it. However, i think there is a catch here :)

To be clear my main reason in this discussion is to make U++ API the most pleasant to use as possible. This is not about criticizing some past decisions. We are all here together and we would like to help and make U++ even better.

Klugier