Subject: Re: Issues using 2015.2 version
Posted by Oblivion on Sat, 23 Jan 2016 21:59:27 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Sat, 23 January 2016 13:08I think this ideal case for pick(), as you pick up content of return value (leaving it empty), which is discarded anyway right away.

In programming you always should be sort of aware how you use the computer memory.
In C++ you have great control over it, actually just by reading source you can tell quite exactly, how the memory is used.

Vector is built of two parts, the Vector container data like size, capacity and pointer to the data itself; this part is of constant memory size, and is cheap to allocate/copy/destroy, so it's often instantiated on the actual scope stack.
And then there're the data itself, which are allocated on the heap memory, as there may be lot of them, and the pointer to the heap is stored inside the container inner structure.

pick() is sort of "copy constructor", which will copy the size/capacity/pointer from one Vector to the other Vector container, and it will disconnect and invalidate the old container's data pointer. Effectively "moving" the Vector without copying millions of data, only the small constant size Vector container inner structure is copied.

clone() is full copy constructor, which will allocate another (second) block of heap memory for the actual data inside the new Vector instance, and then will copy the actual data from one block of heap memory to the new one.
The old instance of Vector remains fully operational. This is performance expensive operation, and you don't want to do it, unless you actually need a second copy of data.

So when you have a cheap way to move the Vector data around with "pick", you can instantiate Vector inside function, fill it up with data, return the temporary function instantiated Vector (to be released on the very next line after return from function), and pick the data payload to caller's Vector instance, saving them from release, but not copying them one by one.

If you are still confused, which of those two used, use always "pick" in debug mode. In case you mess it up, and access the picked container after, it will crash with some assertion, that you tried to access data from picked Vector, then you can decide if you need a copy, or you accidentally access old Vector instead of the new one.

In case of function returning Vector there's no reason for "clone()", as the source instance is destroyed right away in the function call clean up.

+1

This clear and simple explanation of the pick() and clone() taking vector as an example, should be included in the U++ docs, really. :)

Regards,