Subject: Note about how classic OOP with C++ fails efficiency
Posted by mr_ped on Wed, 24 Feb 2010 11:49:21 GMT
View Forum Message <> Reply to Message

Interesting reading (at least for me it was worth a read):
http://solid-angle.blogspot.com/2010/02/musings-on-data-orie nted-design.html

Subject: Re: Note about how classic OOP with C++ fails efficiency
Posted by tojocky on Wed, 24 Feb 2010 12:44:50 GMT
View Forum Message <> Reply to Message

The multi-threading did not resolve this problem?

mr_ped wrote on Wed, 24 February 2010 13:49Interesting reading (at least for me it was worth a read):
http://solid-angle.blogspot.com/2010/02/musings-on-data-orie nted-design.html

Subject: Re: Note about how classic OOP with C++ fails efficiency
Posted by mr_ped on Wed, 24 Feb 2010 13:15:49 GMT
View Forum Message <> Reply to Message

It actually makes things worse in this aspect (slow memory)... how it should resolve it?

Subject: Re: Note about how classic OOP with C++ fails efficiency
Posted by Mindtraveller on Thu, 25 Feb 2010 07:52:40 GMT
View Forum Message <> Reply to Message

Very interesting, thank you.
1. I'm afraid C++ with C++0x standard is moving into something different direction. Compiler will support transparent vectorization with OOP, but not for C++ and not in our Universe.
2. It looks like future architectures will have highly asynchronous multi-core CPU and still good old RAM. This should increase effective memory latency in cycles per 1 CPU core from 200:1 (which is actual for modern x86 PCs) to 1000:1 or more.

Subject: Re: Note about how classic OOP with C++ fails efficiency
Posted by mr_ped on Thu, 25 Feb 2010 08:45:13 GMT
View Forum Message <> Reply to Message

The good thing is, that the absolute speed of everything is going up, so unless you are game developer dealing with GBs of data per second, you can pretty much stick to "good old OOP" in C++, and unless you do something very stupid, you can safely ignore the 1000:1 RAM latency

and the app will be still fast enough.

Basically I still do believe the OOP "by book" approach allows for high abstraction, thus leading to lean source code which is easy to maintain and even reuse sometimes, although the performance is suboptimal.

I'm afraid DOD principle will generally lead to slightly more complex code. Then again the real life example from graphics shaders shows the DOD can lead also to lot of simple pieces of code, which is not lot more difficult to manage then shorter general OOP class doing all the stuff in one place, for some people it may be even simpler to manage.

I can imagine some cases where DOD will actually give you a mental shortcut to better classes with simpler interface (where going there trough OOP would take 3-4 versions at least), leaner code and better performance, so I have to be more aware of it and catch such cases early and use it to my advantage.

Subject: Re: Note about how classic OOP with C++ fails efficiency
Posted by Mindtraveller on Thu, 25 Feb 2010 11:09:47 GMT
View Forum Message <> Reply to Message

I must agree only at some part. Because, finally, we have to answer the question: "what are the programs we create?" Usually, if we talk about desktop apps, it is most-of-time-sleeping finite state machine. Each time user does action, machine awakes and does some job. What is this job about? It is always about doing some simpler task with a number of entities. That is why each language vastly depends on efficiency of it's containers. Cycling through containers is the most common task we do. Of course, efficiency depends on WHAT we do with elements, but cycling through is too a frequently executed thing.
