Subject: Re: The problem with 'Null'
Posted by cbpporter on Thu, 19 Mar 2009 07:45:23 GMT
View Forum Message <> Reply to Message

gridem wrote on Thu, 19 March 2009 09:04But using such approach the programmer should distinguish between init and non init state. One of the possible solution: apply 'Null' to the fields and than check by using IsNull.
I think that with U++ design there is no such thing as init and non-init state. Pretty much all classes are in "init" state after the call of a constructor, even the default one. Such objects are initialized and ready to use, but generally hold no extra information. A String() will have zero length, a Vector() will have zero elements, a Button() will have no picture, text and other properties that diverge from default, but otherwise is ready to be inserted into a parent.

So basically the default constructor creates object in an already prepared state. I think the reason that we use default constructor in this way rather than constructor with parameters to set all the meaningful data is largely related to the intrinsic construction rules of C++. Since we don't put everything on the heap, the objects construction can't be deferred to the allocation moment and is constructed at the runtime point equivalent to the declaration. At this point you often don't have all the information required to call a constructor with parameters.

There are of course exceptions. Things like Size and Point will not be initialized by default construction and can result in bugs if the user is not aware of this fact. Here denoting the absence of an initialization would be useful, but I think these classes are supposed to be lightweight and this is why they are not initialized. Adding unfertilized state detection would make them more heavy weight and less efficient. Having a vector of 10 points would initialize all of them to default, and in most cases, you would reinitialize them in code again. By leaving them uninitialized you only get the meaningful initialization, but you risk bugs if you forget it.

As for the One<Vector <T> > solution, I think is not a very good one. Not only do you have to test if the Vector is not Null and loose the guarantee that Vector is initialized, but you also increase the level of indirection. Such tricks might look like something good, but my experience tells me that if you are going to write programs with hundreds of thousands of line of code and use less straight forward solutions, you'll end up causing more problems than you solve.

Wouldn't a call to IsEmpty or something equivalent on you return value solve the problem? Maybe I'm not understanding exactly what you are trying to achieve. Maybe if you can give a real world example where U++ initialization scheme is not working out and you need to determine initialization status, then I could offer a better solution.