Subject: Re: Core chat...
Posted by mdelfede on Thu, 25 Oct 2007 21:47:46 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 25 October 2007 23:38

Well, performance is nice, but really not that important.

What IS imporant is that there is only ONE PLACE where instance of (possibly) non-copyable object can exist.

I don't get the point...
Quote:
Quote:

```
array<int> a, b;
a.At(1000) = 1;
b = a;      <== b and a share same memory area
a[10] = 2;  <== here an automatic deep copy, ok
b[10] = 2;  <== here no deep copy, just array access, ok
```

Note that above is impossible to implement reliably in C++ (as long as you want read operator[] access to perform no copy at all).

Yes, you got the true caveat of my way.... now maybe you understand *why* I do miss __property construct in c++....

Quote:

Sure. Anyway, the real point of pick is here:

```
Array<Ctrl> CreateWidgets()
{
   Array<Ctrl> x;
   ...
   return x;
}
```

uh ? My Array class behaves exactly as yours, here...

```
Array<Ctrl> CreateWidgets()
{
   Array<Ctrl> x;  <== here, a single reference to memory object
```

```
    ...
    return x; <== here, for a while, 2 references to THE SAME memory object
}

ctrls = CreateWidgets() <== here, the first reference is destroyed, leaving a single reference in
ctrls
```

In your pick_ behaviour, you have a single reference ever to a single memory object. In my case, I have just for a while 2 references to a single memory object, then the first one is released leaving the same result as yours.

Max