Subject: Re: Core chat...
Posted by mdelfede on Wed, 24 Oct 2007 18:24:03 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 24 October 2007 18:23
Well, I guess there is always instrinsict conflict between performance, convenience and safety.

Speaking about performance, I agree... it shouldn't traded with convenience. With safety.... hmmm... there I have some reserve.
But, in modern compilers, doing a 3 line construct or an equivalent 1 liner At() should be the same in terms of performance.
BTW, I really see few practical usage of At(); normally (about 100% of cases) you know in advance your needed array size, and, if you don't, I don't really see the point of increasing (and such copying the whole stuff) the array on a 1 element basis; se that example :

```
Array<int> a(1);
for(int i = 2, i < 1000; i++)
  a.At(i) = i;
```

of course that works, but that means, in the worst case of a badly written Array code, 998 realloc() calls, with 998 buffer copy, memory releases and allocations, ecc ecc.
In a better and more realistic case, if the array is grown in chunks of 10 elements, you'd still have to do it 98 times. That's no efficiency neither good code.
Of course there's a better solution (here I added 1 member functions to Array class) :

```
Array<int> a(1);
for(int i = 2, i < 1000; i++)
{
  a.CheckSize(i, 1000);
  a[i] = i;
}
```

Where the CheckSize(minSize, increment) is a member function that checks the actual/required size of array and, if too small, make it to grow of a specified size (here 1000).
That'd be an huge preformance gain with a small line of code added.

Quote:
"At", and in fact, the whole concept of "inplace creation" (instances get created in container, reference is returned) is nothing new, it is part of U++ design since the very beginning and proved to be very useful.

In any case, I believe I would make more bugs in "unrolling"

```
a.At(x).At(y) = ...
```

code than using such utility method.

Here I'd let the At() the sole purpose of element accessor and put somewhere else the array resize. I don't see nothing bad on a.At().At().At(), as they don't change array sizes.

Ciao

Max
