Subject: Re: A [TreeCtrl] bug
Posted by mr_ped on Tue, 04 Mar 2008 11:38:38 GMT
View Forum Message <> Reply to Message

I didn't check TreeCtrl source nor do I know how it works, but if some memory spot is marked as "free", the value can be there undefined?
So if you find there the key you are searching for, it does not exclude the case of that particular key to be also found on some other spot, where the item[i].free == false?

In such case your optimization will break the result.

And it contains duplicate code... I would refactor it anyway to:

```
int    TreeCtrl::Find(Value key)
{
 for(int i = 0; i < item.GetCount(); i++)
  if (Get(i) == key)
   return item[i].free ? -1 : i;
 return -1;
}
```

Still I think it may be broken in the case of some valid key being also stored second time in some "free" item. (But I'm too lazy to check TreeCtrl source and learn how it works and how it is used.)

Quote:key = Null is nessary. for memory release.
Why? I don't see any connection between those two in C++.
Although I admit it's a good practice in C++ to always null your pointers after you released the allocated memory they are pointing to, so you don't access the freed memory by accident and rather get exception from null pointer, but actually it's "memory release for key = null", not "key = null for memory release", the reason and the consequence are the other way around.