Subject: Re: revised ownership change
Posted by Lance on Thu, 28 Apr 2011 19:45:59 GMT
View Forum Message <> Reply to Message

Good job.

Except the template creator should be kept. It's simulating the Array::Create<ChildType>() interface. Look at it this way, a Ctrl is a special type of container: it references some of its children while owning (responsible for their destruction) some others. Since allowing library user to change the owned flag is regarded as unsafe, and use of new/delete by library users is generally discouraged, it makes sense to have a similar Create/Detach interface like does Array.

The point of the template creator (NewChild, AddOwned, CreateOwned or whatever we may choose to call it) is to eliminate the necessity of asking end user to supply a new'ed object, instead of:

```
parent.AddOwned(new Button());
```

now use

```
parent.CreateOwned<Button>();
//
// similar to
// anArray.Create<Button>();
//
```

Yes, any access to owned flag should be protected. That's my true intention. Thanks for fixing it.

I still think that to expose Detach to public scope is not a good idea. It should be protected and only accessible through inheritance. Even though it works in common/reasonable senario:

1. a Ctrl is Detached from its parent:
In any case, the Ctrl is removed from its parent's children list. If the Ctrl has owned flag set, a pointer to it will be returned so that user code will be responsible for its eventual destruction; if its a normal stack resident Ctrl. a NULL pointer will be returned to signal the user not to try to delete it afterwards.

2. the user delete's the detached Ctrl after finished using it. No problem, expected scenario.

3. the user decide to add the ctrl to another Ctrl as child by way of Add(), AddChild(), etc. Since the owned flag is correctly set, the new parent will be responsible for its destruction.

So it seems no hole is introduced. But here Add(), AddChild() becomes ambiguous to the user. The user may form an impression that Add()/AddChild() can be supplied with any new'ed Ctrls without the user to worry about their eventual destruction. To avoid this confusion, maybe it's best to simply hide it from library users.