Subject: Re: widget lifetime
Posted by mirek on Sun, 02 Apr 2006 13:46:48 GMT
View Forum Message <> Reply to Message

[quote title=gprentice wrote on Sun, 02 April 2006 05:31]
The UPP website overview page has this

 instead of
```
struct MyDialog {
    Option *option;
    EditField *edit;
    Button *ok;
};
```

we are using:

```
struct MyDialog {
    Option option;
    EditField edit;
    Button ok;
};
```

Even more important, lifetime of these widgets does not depend
on the life cycle of MyDialog GUI - MyDialog can be closed
or not yet open, but attributes of widgets are accessible all
the time.

Quote:
I don't follow the argument here. The pointers in the first case can be assigned in the MyDialog constructor and their pointee destroyed in the destructor, giving the same lifecycle as the UPP strategy - or is there some technical reason why toolkits that do the pointer/heap strategy can't do this (which toolkits do the pointer/heap thing - Delphi/BCB/wxWidgets ??).

First those are two distinct (albeit related) features (no pointers and lifetime not limited by toolkit).

Second, yes, you could do destruction in destructor, but that would be a lot of code to write. In fact, you CAN do that if you want with U++.

As for other toolkits - any other C++ toolkit known to me REQUIRES widgets to reside on heap. I most cases, the reason is exactly because it is a lot of work to delete them manually, so toolkit takes care about deleting of widgets (therefore they MUST be on the heap). And of course, as there needs to be a point of deletion, widgets are usually deleted when GUI lifecycle ends (dialog closed). Kind of architectural deadlock, which we refused to share

Quote:
So why don't other toolkits do it the UPP way - does it make the static size of a program bigger?

Somebody had to be first

Also besides argument already presented, worth noting is that this architecture is only possible with C++. Other toolkits often follow non-C++ widget architecture models (e.g. taken from C or Java...).

Quote:
Where do UPP dialogs normally live - on the stack or on the heap?? - in examples we have MyApp().Run() - which goes on the stack, however stack space is more limited than heap space ??

You have to difer "inteface" and "implementation" part of thing. "interface"-like live usually on stack, however "implementation" often uses the heap (usually via NTL containers and String/Value and other concrete classes).

Quote:
The UPP web page says the toolkit neither creates or destroys widgets - that means the application does it all ... - so does that mean a UPP application does lots of new'ing and deleting ??

Have you seen some new or delete in examples or TheIDE sources?

Most of real heap deletes is in fact performed by NTL and String implementations. However, at interaface level, using pointers to manage heap resources is a "bad practice" and most of client code does not have to care about heap.

Mirek