Subject: widget lifetime
Posted by gprentice on Sun, 02 Apr 2006 09:31:40 GMT
View Forum Message <> Reply to Message

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.

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 ??).

The pointer/heap way is more code and has the exception safety problem I guess - the destructor isn't called if the constructor didn't complete - means having smart pointers I guess. Hmmm - so is that why these pointers can't be assigned in the MyDialog constructor (unless they're smart pointers) ???

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

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 ??

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 ??

Graeme
[BTW (fudadmin - are you reading this?) - does preview message work - how do you get back to your message after previewing it?]