Subject: Modal vs non-modal window
Posted by NeonN on Wed, 03 Oct 2007 18:13:42 GMT
View Forum Message <> Reply to Message

Hi, I'm new here. I found Ultimate++ few months ago... but until last week, i had no time to try it. And I have to say... I am starting to like it more and more ...

But there's a problem... I do not understand how to open dialog window (and block the windows behind) and how new main window or new dialog window without blocking the app...

I tried this

```
void ToDo2::AddProjectDlg()
{
 static  WithAddProjectLayout<TopWindow> * opened = NULL;
 if (opened != NULL)
 {
  opened->TopMost(false,false);
  return;
 }
 bool * sent = new bool;
 WithAddProjectLayout<TopWindow> * layAddProject = new
WithAddProjectLayout<TopWindow>;
 opened = layAddProject;
 CtrlLayoutOKCancel(*layAddProject, "Add Project");
 layAddProject->Open();
 *sent = false;
 while (!(*sent))
 {
  switch (layAddProject->Execute())
  {
   case IDOK:
    if ((String) layAddProject->esProjectName.GetData() != "")
    {
     AddProject(AsString(layAddProject->esProjectName.GetData()),
AsString(layAddProject->deDescription.GetData()));
     *sent = true;
    }
    else
    {
     PromptOK("Project name is empty!");
    }
    break;

   case IDCANCEL:
   case IDEXIT:
    *sent = true;
  }
```

```
 }
 delete sent;
 delete layAddProject;
 opened = NULL;
}
```

```
GUI_APP_MAIN
{
 ToDo2 main_abc;
 main_abc.Sizeable().Zoomable().Run();
}
```

This is doing exactly what i need. Open window... if allready opened, just get it to the top of the screen (checked with EventLoop())... check some values entered by user and if everything is ok, use them (call some function) and close the window.
But how to open it as independet window or independent main window (show it on taskbar)?

I gues I have to use Execute() to check the dialog status... but Execute() blocks the main windows behind ... and EventLoop() doesn't return any value to check.

Subject: Re: Modal vs non-modal window
Posted by mrjt on Thu, 04 Oct 2007 11:08:24 GMT
View Forum Message <> Reply to Message

Hello and welcome!

I assume you've already decided against a seperate class for the window, so what I would do is something like this:

```
One<AWindow> wnd;
void OpenAWindow() {
 if (!wnd) {
  wnd = new AWindow();
  CtrlLayout(*wnd, "A Window");
  wnd->ok <<= THISBACK(AWindowOK);
  wnd->WhenClose = wnd->cancel <<= THISBACK(AWindowClose);
  //wnd->Open(this);
  wnd->OpenMain();
 }
 else
  wnd->TopMost(false, false);
}
```

```
void AWindowOK() {
 if (wnd->Accept()) {
  // Do Something with data
  AWindowClose();
 }
}
void AWindowClose()  { wnd.Clear(); }
(all members of another class)
```

This allows the window to be displayed non-modally. By changing OpenMain to Open(this) you will get a child window (always above parent). It's possible that Mirek might come along and tell me I've got this wrong though

Also, if you didn't care about memory release after closing (if it's a small app I wouldn't bother), you could just add the line:AWindow &wnd = Single<AWindow>()to all the functions you need to access the window, and remove the member variable.

Hope that helps.
James

edit: Forgot AWindowClose

Subject: Re: Modal vs non-modal window
Posted by NeonN on Thu, 04 Oct 2007 16:59:43 GMT
View Forum Message <> Reply to Message

Thanks for response...
I dont't know if it is better to put the window in separate class or just in the existing one of the main window. I thought there is an option how to do everything (open window, check values on OK button click...) in a single class member. No other callbacks... one dialog in just one function. That is why I was tring to use CtrlLayoutOKCancel and:

```
*sent = false;
while (!(*sent))
{
 switch (layAddProject->Execute())
 {
  case IDOK:
...
...
...
```

And as I am thinking about it I guess it would be better to have some bigger or more important windows in a separate class. Could you pls tell me some pros and cons?

I was playing with it today again and a can't find any difference between Open(), Open(this) and

OpenMain() (except OpenMain() makes the new window visible on taskbar). Non of them block the main window behind...

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

#define LAYOUTFILE <WindowTest/WindowTest.lay>
#include <CtrlCore/lay.h>



class WindowTest : public WithLayout1<TopWindow> {
 private:
  void onButton1Click();

 public:
  typedef WindowTest CLASSNAME;
  WindowTest();
};

void WindowTest::onButton1Click()
{
 WithLayout2<TopWindow> * wnd = new WithLayout2<TopWindow>;
 CtrlLayoutOKCancel(*wnd, "Title2");
 //wnd->Open();
 //wnd->Open(this);
 wnd->OpenMain();
}

WindowTest::WindowTest()
{
 CtrlLayout(*this, "Title1");
 Button1 <<= THISBACK(onButton1Click);
}

GUI_APP_MAIN
{
 WindowTest().Run();
}
```

EDIT:
I just find out that new windows opened inside the app with OpenMain() can overlay the main window... but windows opened with Open() can't... but no option disable the main windows. But when I Execute() or Run() the window, the main window is disabled always ...
What am I doing wrong? Or am I so stupid?

Subject: Re: Modal vs non-modal window
Posted by NeonN on Thu, 04 Oct 2007 18:08:15 GMT
View Forum Message <> Reply to Message

Ok, I think i got what i needed ... I thought out the mrjt's post again and used the THISBACK system of dealing with the OK/Cancel bottons instead of CtrlLayoutOKCancel().

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

#define LAYOUTFILE <WindowTest/WindowTest.lay>
#include <CtrlCore/lay.h>

class WindowTest : public WithLayout1<TopWindow> {
 private:
  WithLayout2<TopWindow> * wnd;
  void onButton1Click();
  void onExitWindow2();
  void DealWithWindow2();

 public:
  typedef WindowTest CLASSNAME;
  WindowTest();
};

void WindowTest::onButton1Click()
{
 wnd = new WithLayout2<TopWindow>;
 CtrlLayout(*wnd, "Title2");
 //wnd->Open();
 //wnd->Open(this);
 wnd->OpenMain();
 this->Disable();
 wnd->WhenClose = wnd->ok <<= THISBACK(DealWithWindow2);
}

void WindowTest::DealWithWindow2()
{
 this->Enable();
 delete wnd;
}

WindowTest::WindowTest()
{
 CtrlLayout(*this, "Title1");
 Button1 <<= THISBACK(onButton1Click);
}
```

```
GUI_APP_MAIN
{
 WindowTest().Run();
}
```

It's working but I don't know if it is ok... or if there is an easier/better option?
Next question is the pros and cons of having the new dialog window in a separate class (I am talking about window with about 10-20 edits or other components to fill + saving it all to DB).

I would be glad of any response ... and sorry for stupid questions . I am porting one app from PHP and I am not used to use classes or any GUI objects...

Subject: Re: Modal vs non-modal window
Posted by exolon on Fri, 05 Oct 2007 09:55:53 GMT
View Forum Message <> Reply to Message

Just out of curiosity, why did you do:
bool * sent = new bool;

and dereference it all the time and eventually have to delete it?
I would recommend against creating objects dynamically if you don't really need to do it... there's surely no good reason to create inbuilt types like bool and int on the heap, since the pointer on the stack is at least as large as a bool and dereferencing it is ugly and you can make mistakes (including forgetting to delete, or deleting twice).

Use the stack.

Subject: Re: Modal vs non-modal window
Posted by mrjt on Fri, 05 Oct 2007 10:09:51 GMT
View Forum Message <> Reply to Message

There are no stupid questions

There are lots of different ways of handling windows in Upp, and choosing which method to use is dependent on the complexity and type of the window and application as well as personal preference.

Generally I would use seperate classes if the window has more complex behaviour, such as values in one field changing allowable values in others, or Option controls that disbale/enable fields etc. You could do all this from the parent window but it would become untidy and complex, especially if you have lots of them.

If you can make your window a modal dialog (one that blocks other windows) it becomes easier to

do more complex windows without a seperate class because it can all be handled in one function. A good example of this that I often use as a reference is void Ide::SetupFormat() in the ide package (ide/Setup.cpp) because it handles quite complex data (look at CtrlRetriever in particular, this might be useful to you) and you can easily compare it to the running version in TheIde.

Once you have non-modal windows it is more difficult becuase you have to seperate the OK button handling, but it really depends on the types of windows you are using. If for instance you have many windows that are just data entry you may be able to simplify the process so that seperate classes are unnecessary by creating a single class that inherits from TopWindow and behaves in a general way. For example, this:

```
class DataEntryWindow : public TopWindow
{
public:
    typedef DataEntryWindow CLASSNAME;

    // Sets the table name to update
    DataEntryWindow &SetTableRecord(String table, int recordid);
    // Links a DB field with a Ctrl
    DataEntryWindow &SetDataSource(Ctrl &source, String field, Value intial_value, int datatype);

    OnOK()            { if (Accept()) { Commit(); OnCancelClose();} }
    OnCancelClose()    { delete this; }
private:
    Commit(); // Builds SQL string using datasources and commits it to DB
};
```

could form the basis of a data enrty window that can be launched from a single function and then forgotten about:

```
void NewEmployee()
{
    WithEmployeeLayout<DataEntryWindow> *wnd = new
WithEmployeeLayout<DataEntryWindow>();
    //Set tablename, datasources etc
    wnd->Open(this);
}
```

Note that I've never used Upp for SQL, so I'm not sure how well this would work, but something similar should be possible. You may also be able to do something with Serialize().

Other than all that (I got a bit carried away I think, I really should be doing some work) you just need to play around, Upp is very versatile once you know what your doing. Reading the source code (both CtrlLib and ide) helps as it was written by very good programmers.

Feel free to ask anything else, answering questions is always good to get the brain working in the morning

James

Subject: Re: Modal vs non-modal window
Posted by NeonN on Fri, 05 Oct 2007 11:47:14 GMT
View Forum Message <> Reply to Message

exolon wrote on Fri, 05 October 2007 11:55Just out of curiosity, why did you do:
bool * sent = new bool;

Thanks for a note... I guess I wasn't thinking about the code deeply ..

mrjt:
I guess you're right. Separating more complex windows is a good idea. And for small dialogs it's reasonable to use just:

```
void HomeBudget::NewCategory()
{
 AddNewCat dlg;

 if(dlg.Execute() == IDOK)
 {
  ...
 }
}
```
(HomeBudget example)

I looked at the CtrlRetriever and it seems to be a good way how to handle Option dialogs... thanks for a tip. I'll try it as soon as possible.
And the idea of creating a dialog class with some basic functions to manipulate the window and it's data crossed my mind yesterday. But first I'll have to explore some features of Ultimate++ ... I am getting into it slowly (but surely).

Anyway...for now thanks for all the answers ... Ultimate++ is great toolkit and this forum is awesome. Ultimate++ gained another happy user ...

Subject: Re: Modal vs non-modal window
Posted by mrjt on Fri, 05 Oct 2007 12:47:32 GMT
View Forum Message <> Reply to Message

Glad to hear it

ArrayCtrl is also very useful for lots of Option controls.

Subject: Re: Modal vs non-modal window
Posted by mirek on Sat, 06 Oct 2007 15:59:30 GMT
View Forum Message <> Reply to Message

mrjt wrote on Thu, 04 October 2007 07:08

This allows the window to be displayed non-modally. By changing OpenMain to Open(this) you will get a child window (always above parent). It's possible that Mirek might come along and tell me I've got this wrong though

Only a little terminology fault:

These windows are really called "owned", not "child".

Also, Open() picks "last active window" of application as its owner (which usuallly is the right window to use..). That is why there has to be OpenMain (which does not pick anything, makes it the "main" window).

Mirek

Subject: Re: Modal vs non-modal window
Posted by mirek on Sat, 06 Oct 2007 16:07:56 GMT
View Forum Message <> Reply to Message

mrjt wrote on Fri, 05 October 2007 06:09

```
class DataEntryWindow : public TopWindow
{
public:
    typedef DataEntryWindow CLASSNAME;

    // Sets the table name to update
    DataEntryWindow &SetTableRecord(String table, int recordid);
    // Links a DB field with a Ctrl
    DataEntryWindow &SetDataSource(Ctrl &source, String field, Value intial_value, int datatype);

    OnOK()            { if (Accept()) { Commit(); OnCancelClose();} }
    OnCancelClose()    { delete this; }
private:
    Commit(); // Builds SQL string using datasources and commits it to DB
};
```

could form the basis of a data enrty window that can be launched from a single function and then forgotten about:

```
void NewEmployee()
{
    WithEmployeeLayout<DataEntryWindow> *wnd = new
WithEmployeeLayout<DataEntryWindow>();
    //Set tablename, datasources etc
    wnd->Open(this);
}
```

Not that this is only a good idea if you have unlimited number of peer windows (like in UWord), preferably even main peer windows.

For modeless things, usually it is better to just make those subdialogs a member variables (no pointers); to open and close them as needed, no news and deletes involved. This also has the advantage that the content of dialog is accessible in the containing class, so there is often no need to transfer widget data to variables and back...

Mirek

Subject: Re: Modal vs non-modal window
Posted by NeonN on Sun, 07 Oct 2007 16:33:58 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 06 October 2007 17:59
Also, Open() picks "last active window" of application as its owner (which usuallly is the right window to use..). That is why there has to be OpenMain (which does not pick anything, makes it the "main" window).

Thx, a lot. I was wondering what is the difference between Open() and Open(this) ... now I get it.

luzr wrote on Sat, 06 October 2007 18:07
For modeless things, usually it is better to just make those subdialogs a member variables (no pointers); to open and close them as needed, no news and deletes involved. This also has the advantage that the content of dialog is accessible in the containing class, so there is often no need to transfer widget data to variables and back...

I read it once before somewhere on this forum... is there any other reason to don't use 'new' for dialogs? If I use 'new', the window doesn't disappear after it's init class-member function is done and the widgets are cleared after closing the window. But I don't know if it's OK to use Ultimate++ this way ...

Subject: Re: Modal vs non-modal window
Posted by mirek on Sun, 07 Oct 2007 17:37:50 GMT
View Forum Message <> Reply to Message

NeonN wrote on Sun, 07 October 2007 12:33
I read it once before somewhere on this forum... is there any other reason to don't use 'new' for dialogs?

Well, the fundamental believe of U++ paradigm is to avoid "delete" whenever possible (more than

one delete per 1000 lines of code is just too much . Usually, and especially in this case, it simply leads to more elegant and easier to maintain code.

Quote:
If I use 'new', the window doesn't disappear after it's init class-member function is done and the widgets are cleared after closing the window

Do not quite understand this...

Quote:
But I don't know if it's OK to use Ultimate++ this way ...

OTOH, if you insist on more complicated code, there is not fundamental problem with "new"/"delete" and U++

Mirek

Subject: Re: Modal vs non-modal window
Posted by NeonN on Sun, 07 Oct 2007 18:32:05 GMT
View Forum Message <> Reply to Message

For example:
I have one window with button. The button has assigned callback to function that creates another window...

If I just do this:

```
void ClassName::InitDialog()
{
 WithLayout<TopWindow> laySomething;
 CtrlLayoutOKCancel(laySomething, "Something");
 laySomething.OpenMain();
}
```

... the window disappear after the function is over. I'd have to Execute() the window or ... (I don't know... I'm new in C++ and Ultimate++. I'm trying to understand how to work with the window - what is the best method...)

But when I do this:

```
void ClassName::InitDialog()
{
 laySomething = new WithSomethingLayout<TopWindow>;
 CtrlLayoutOKCancel(*laySomething, "Something");
 laySomething->OpenMain();
}
```

... the window is still opened. I can do anything with the window because the pointer to it is declared in the private part of class but the memmory is allocated only when I need it. And when I close the window, the widgets are automatically cleared (destroyed).
So that's why i don't understand why is using new/delete less alegant or less easier... and why I shouldn't do this...

Subject: Re: Modal vs non-modal window
Posted by mirek on Sun, 07 Oct 2007 21:46:33 GMT
View Forum Message <> Reply to Message

class ClassName {
...
  WithLayout<TopWindow> laySomething;
...
};

NeonN wrote on Sun, 07 October 2007 14:32and why I shouldn't do this...

Well, e.g., when you close the main window, what happens to owned ones? Making owned windows proper class members solves this problem implicitly.

Mirek

Subject: Re: Modal vs non-modal window
Posted by NeonN on Mon, 08 Oct 2007 07:14:10 GMT
View Forum Message <> Reply to Message

Thx, i get it now... i wasn't thinking about that (making the dialow window class member). But I'm not very exited opening the window this way because it's IMHO wasting free memmory... the dialog window is allocated all the time when main window is running...

Maybe it's not so important for me because the app i am porting will be using at most 5-10 small dialogs... but anyway, i don't like it at all ....

Subject: Re: Modal vs non-modal window
Posted by mirek on Mon, 08 Oct 2007 08:52:26 GMT
View Forum Message <> Reply to Message

NeonN wrote on Mon, 08 October 2007 03:14Thx, i get it now... i wasn't thinking about that (making the dialow window class member). But I'm not very exited opening the window this way because it's IMHO wasting free memmory... the dialog window is allocated all the time when main

window is running...

Maybe it's not so important for me because the app i am porting will be using at most 5-10 small dialogs... but anyway, i don't like it at all ....

Yep, that is, in theory, a valid concern.

Anyway, how big is subdialog? Single widget occupies 100-200 bytes, so it is usually less than 5KB. People tend to waste much more in other parts of code without even knowing

Another important issue is that you will get access to the dialog, all its settings and values, for free. With new, you would in most cases needed to store a pointer to subdialog anyway.

OTOH, if the memory really is concern, you can use something like

```
class ClassName {
...
  One< WithLayout<TopWindow> > laySomething;
...
};

void ClassName::InitDlg()
{
...
  laySomething.Create();
...
}
```

In this case, data will be allocated on demand, while still keeping the advantage of deterministic cleanup.

Subject: Re: Modal vs non-modal window
Posted by NeonN on Mon, 08 Oct 2007 12:58:48 GMT
View Forum Message <> Reply to Message

Thx for the One<> tip... it seams really useful.

luzr wrote on Mon, 08 October 2007 10:52
Another important issue is that you will get access to the dialog, all its settings and values, for free. With new, you would in most cases needed to store a pointer to subdialog anyway.

Yes, the pointer to the dialog would be a member of main class (and the dialog window would be new'ed in the window-init function)... so I could access to the widgets and values of them in the whole class. But now, the One<> seems as a better option. Thx again...

The app will contain approximately 5-10 dialogs... some of them are just few edits put together but few will be more complex (2-4 ArrayCtrl with data selected from DB, edits, droplists, buttons - all in 2-3 tabs)... so I guess it's more than few kB .