Subject: Virtual functions versus callbacks
Posted by koldo on Tue, 05 Jan 2010 08:59:08 GMT
View Forum Message <> Reply to Message

Hello all

To let the program do actions after events I have seen there are two options in Upp:

- To use "virtual" functions
For example to get the "enter" key in an EditField it is necessary to do a derived class
that catches virtual bool Key(dword key, int rep);

- To use callback
For example to do something after clicking in an ArrayCtrl it could be done by assigning a callback function to "WhenLeftClick".

From the user (programmer) point of view it seems better callback functions as you do not need to do derived classes every time you need a certain response after a Ctrl event.

Personally I stronger prefer callback but, what do you think about it ?

Best regards
Koldo

Subject: Re: Virtual functions versus callbacks
Posted by unodgs on Tue, 05 Jan 2010 09:36:50 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 05 January 2010 03:59Hello all

To let the program do actions after events I have seen there are two options in Upp:

- To use "virtual" functions
For example to get the "enter" key in an EditField it is necessary to do a derived class
that catches virtual bool Key(dword key, int rep);

- To use callback
For example to do something after clicking in an ArrayCtrl it could be done by assigning a callback function to "WhenLeftClick".

From the user (programmer) point of view it seems better callback functions as you do not need to do derived classes every time you need a certain response after a Ctrl event.

Personally I stronger prefer callback but, what do you think about it ?

Best regards
Koldo

In my upp sources EditField has WhenEnter callback. I guess it could be added to the main tree as now I can see I'm not the only one who needs it

---

Subject: Re: Virtual functions versus callbacks
Posted by mrjt on Tue, 05 Jan 2010 16:55:56 GMT
View Forum Message <> Reply to Message

There are two additional options you missed out:

3- Catch unhandled key presses in the Ctrls owner/parent. ie:

```
virtual void Key(dword key, int count)
{
    if (key == K_RETURN && password_field.HasFocus())
       login.PseudoPush();
    else
       return false;
    return true;
```

}Since you'd have to declare a new function to handle a callback anyway this isn't really any extra code.

4- Create a simple template to automate callback functionality:

```
template <class T>
struct WithCallbacks : public T {
   Callback WhenLeftDown

   virtual void LeftDown(Point p, dword keyflags) { WhenLeftDown(); }
}
...
WithCallbacks<ArrayCtrl> array;
```

Given all the possible methods it doesn't seem worth retro-actively adding callbacks everywhere.

OTOH, the K_RETURN event for EditField is common enough to warrant it's addition. It should be called WhenReturn or maybe WhenKeyReturn to match the key enum though. WhenEnter is too easy to confuse with the focus changing IMO.

Subject: Re: Virtual functions versus callbacks
Posted by koldo on Wed, 06 Jan 2010 08:51:19 GMT
View Forum Message <> Reply to Message

Hello all

The general idea I wanted to share with you is that to use Control events we use or virtual functions or callbacks.

For me callbacks are easier because they do not require to do new classes... just assign the callback to your function.

Is it this way or there are other advantages of virtual functions over callbacks ?

Best regards
Koldo

Subject: Re: Virtual functions versus callbacks
Posted by mrjt on Wed, 06 Jan 2010 10:29:39 GMT
View Forum Message <> Reply to Message

IMO the difference is that Callbacks are intended for communication between classes/Ctrls and are defined by the Ctrl. Callbacks are usually for actions that imply some sort of state change or contextual (wrong word maybe?) function of the Ctrl.

The virtual methods are the way in which a Ctrl interfaces internally with the underlying GUI/OS and is essentially a cleaner replacement of the message loop. The functions sometimes pass additional information that wouldn't be useful or needed externally.

Example of Callbacks:
WhenAction
WhenAcceptEdit

Example of virtual function:
LeftDown
GotFocus

While there are occasions when the developer might requrie some additional comunication from a Ctrl (ie a Callback) that is the exception rather than the rule in my experience and as demonstrated it is not difficult to add.

I think the current way works very well. Adding Callbacks for all events would just expose a lot of internal message stuff that wouldn't be used very often at the expense of making derived Ctrls more difficult.

IMO obviously, only Mirek could give a definitive answer.

Subject: Re: Virtual functions versus callbacks
Posted by koldo on Wed, 06 Jan 2010 10:45:04 GMT
View Forum Message <> Reply to Message

Thank you Mrjt
Koldo

Subject: Re: Virtual functions versus callbacks
Posted by mirek on Wed, 06 Jan 2010 21:09:02 GMT
View Forum Message <> Reply to Message

unodgs wrote on Tue, 05 January 2010 04:36koldo wrote on Tue, 05 January 2010 03:59Hello all

To let the program do actions after events I have seen there are two options in Upp:

- To use "virtual" functions
For example to get the "enter" key in an EditField it is necessary to do a derived class that catches virtual bool Key(dword key, int rep);

- To use callback
For example to do something after clicking in an ArrayCtrl it could be done by assigning a callback function to "WhenLeftClick".

From the user (programmer) point of view it seems better callback functions as you do not need to do derived classes every time you need a certain response after a Ctrl event.

Personally I stronger prefer callback but, what do you think about it ?

Best regards
Koldo
In my upp sources EditField has WhenEnter callback. I guess it could be added to the main tree as now I can see I'm not the only one who needs it

Just to be sure, if callback is empty, Enter is ignored and passed to the parent?

Mirek

Subject: Re: Virtual functions versus callbacks
Posted by mirek on Wed, 06 Jan 2010 21:11:09 GMT
View Forum Message <> Reply to Message

mrjt wrote on Wed, 06 January 2010 05:29
IMO obviously, only Mirek could give a definitive answer.

I would say that

- virtual methods are fine for representing input interface
- callbacks are good for representing output interface