Subject: What way is best to implement Callback for GotFocus/LostFocus?
Posted by jlfranks on Mon, 13 Aug 2007 20:48:56 GMT
View Forum Message <> Reply to Message

We are solving the problem of editing a control in a touchschreen application.

For example, when EditString gets focus, we want to show a keyboard and be able to press buttons in order to modify the content of the control.

The first issue is generalizing the solution so that we can apply this to multiple EditString controls, or even EditDouble, EditInt.

One way is to derive from EditString and over-ride GotFocus() and LostFocus (), along with supporting API to set the callbacks for each.

A different approach is to hook the mouse event, check for the control of interest and with additional API and call the appropriate callbacks for various events/controls.

Which way is more desirable from a design viewpoint? Or, is there
another approach different from these?

--jlf

Subject: Re: What way is best to implement Callback for GotFocus/LostFocus?
Posted by mrjt on Tue, 14 Aug 2007 08:58:24 GMT
View Forum Message <> Reply to Message

You can actually implement this in a general way quite easily, and without having to derive anything.

Ctrl has two methods, ChildGotFocus and ChildLostFocus, that you can overload in the parent window to catch the event. Then you just have to see if an edit ctrl has the focus:

```
 virtual void ChildGotFocus() {
  Ctrl *c = GetFocusChild();
  if (!c || !c->IsEditable()) return;

  if (dynamic_cast<EditField *>(c) || dynamic_cast<TextCtrl *>(c)) {
   // Show keyboard
  }
 }

 virtual void ChildLostFocus() {
  // Unless focus has moved to the keyboard, hide it here
 }
```

You may need to add some other control types, or use
GetFocusChildDeep if you use ArrayCtrls. This is not the only other solution though.

James
