Subject: Focus problem
Posted by Silvan on Tue, 14 Dec 2021 20:23:50 GMT
View Forum Message <> Reply to Message

Problem description:

in the simple test app below I can't unfocus the editstring ctrl so when I use the keyboard to move the character on the string the key pressed goes also on the edit control.

I suppose it is silly but I can't find a simple solution and I don't understand the focus and event mechanism.

Greetings
Thank you
Silvan

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

using namespace Upp;


struct MainWindow : TopWindow {

   Point  p;
   EditString inputtext;

   virtual void Paint(Draw& w)override
    {
 int x,y;

       w.DrawRect(GetSize(), SWhite());

 w.DrawText(p.x, p.y, "#", Arial(30), Red);

 }

   void Close() override
   {
      delete this;
   }

 virtual bool Key(dword key, int count) override
{

      switch (key)
 {
     case K_W:
```

```
          p.y-=1;
        break;
       case K_S:
          p.y+=1;
        break;
       case K_A:
          p.x-=1;
        break;
       case K_D:
          p.x+=1;
        break;
       default:
       //C Statements
       ;
 }

  Refresh();
        return true;
 }

 // Costruttore dove inserisci le inizializzazioni
    MainWindow()
    {

        Title("Test Focus").Zoomable().Sizeable();
  Add(inputtext.TopPosZ(0, 16).HSizePos());
  inputtext <<= "test";

  SetRect(0, 0, 300, 300);
  p.x = 150;
  p.y = 150;
    }
};
GUI_APP_MAIN
{
    (new MainWindow)->OpenMain();
     Ctrl::EventLoop();
}
```

---

Subject: Re: Focus problem

Posted by Silvan on Wed, 15 Dec 2021 13:28:04 GMT
View Forum Message <> Reply to Message

Well, that are my doubts in details:
1) When the program start the focus is set on the EditString Ctrl, it as highlight text and blinking cursor.
That is unusual if confronted with MFC or VCL where usually, if not set, the focus is of the main window.
2) As the situation decrived above if I click on the main windos, outside the EditString Ctrl, I espect that
the focus change to the main windows (no text highlight in the EditString no blinking cursor).
But that doesn't happen.
3) The main window Key event handler return true, so I expect that no other ctrl will proces the same event.
But that is not the case, the key pressed move the draw on the main windows and also write text on the editstring ctrl.
4) Even it disabled in code the EditString still have the blinking cursors.

Thank you if someone would try to give some hints.
Silvan

Subject: Re: Focus problem
Posted by Silvan on Fri, 17 Dec 2021 22:01:34 GMT
View Forum Message <> Reply to Message

Well,
I managed to solve the keyboard event handler adding a new ctrl: ImageCtrl in the area below the EditString Ctrl.
Now I need an example of how to use Function, Event and Gate to overload the member function of my instance of ImageCtrl
to deal with Paint method and Key event.
I searched the upp web site and forum but I didn't find an example even the examples program uses the old THISBACK macro.
I think callback mechanics should be explained better for newby. Thank you.

Subject: Re: Focus problem
Posted by Lance on Sat, 18 Dec 2021 04:39:50 GMT
View Forum Message <> Reply to Message

Hi Silvan:

I manage to create the following example.

THISBACK is legacy code. Event<>, Gate, etc now are alias of Function.

In IDE, Ctrl+Left Click on the Event type name will bring you to its definition, you should see something like this

```
template <typename... ArgTypes>
using Event = Function<void (ArgTypes...)>;

template <typename... ArgTypes>
using Gate = Function<bool (ArgTypes...)>;

template <class Ptr, class Class, class Res, class... ArgTypes>
Function<Res (ArgTypes...)> MemFn(Ptr object, Res (Class::*method)(ArgTypes...))
{
 return [=](ArgTypes... args) { return (object->*method)(args...); };
}
```

BTW, revised code to do what you asked and demonstrate a callback.

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

using namespace Upp;


struct MainWindow : TopWindow {
 Point p;
 EditString inputtext;

 virtual void Paint(Draw& w)override
 {
  int x,y;

  w.DrawRect(GetSize(), SWhite());

  w.DrawText(p.x, p.y, "#", Arial(30), Red);

 }

 void LeftDown(Point p, dword keyflags)override
 {
  SetFocus();
 }

// void Close()override
// {
// // delete this;
```

```
// }

 bool Key(dword key, int count) override
 {

  switch (key)
  {
  case K_W:
   p.y-=1;
   break;
  case K_S:
   p.y+=1;
   break;
  case K_A:
   p.x-=1;
   break;
  case K_D:
   p.x+=1;
   break;
  case K_UP:
  case K_DOWN:
  case K_LEFT:
  case K_RIGHT:
   WhenSuspiciousKey();
  }

  Refresh();
  return true;
 }

 // Costruttore dove inserisci le inizializzazioni
 MainWindow()
 {

  Title("Test Focus").Zoomable().Sizeable().WantFocus();
  Add(inputtext.TopPosZ(0, 16).HSizePos());
  inputtext <<= "test";

  SetRect(0, 0, 300, 300);
  p.x = 150;
  p.y = 150;
 }

 Event<> WhenSuspiciousKey; // accepts a callback with prototype like
 // void keypressed();
};
GUI_APP_MAIN
{
```

```
 MainWindow m;
 m.WhenSuspiciousKey<<[]{ PromptOK("Use upper case W,A,S,D to navigate!");};
 m.Run();
//
// MainWindow *m=new MainWindow;
// m->OpenMain();
//// m->SetFocus();
//// DUMP(m->HasFocus());
// Ctrl::EventLoop();
}
```

BTW, there is nothing wrong with allocating your MainWindow or other Ctrls from stack. But in U++, you don't have to. In your case, there is no reason to do it that way.

---

Subject: Re: Focus problem
Posted by Silvan on Sat, 18 Dec 2021 15:03:45 GMT
View Forum Message <> Reply to Message

Thank you Lance,
understood and it solve all my inizial problem.

And what about if would like to do something like this:

```
 MainWindow()
 {

  Title("Test Focus").Zoomable().Sizeable();
  Add(inputtext.TopPosZ(0, 16).HSizePos());
  Add(panel.VSizePos(18, 0).HSizePos(0, 0));

  inputtext <<= "test";

  ActiveFocus(panel);

  panel.Paint << [&](Draw &w) { w.DrawText(p.x, p.y, "#", Arial(30), Red);};

  panel.Key << [&](dword key, int count) { switch (key) { /
        case K_W: /
         p.y-=1; /
         break; /
        case K_S: /
         p.y+=1; /
         break; /
        case K_A: /
         p.x-=1; /
```

```
          break; /
         case K_D: /
          p.x+=1; /
          break; /
         default: /
          ; /
        } /
       Refresh(); return true; }; /

  SetRect(0, 0, 300, 300);
  p.x = 150;
  p.y = 150;
}
};
```

This code does not compile.... I would override directy the method and event of ImageCtrl panel.
How this is possible?
Thank you
Silvan

---

Subject: Re: Focus problem
Posted by Silvan on Sat, 18 Dec 2021 15:22:39 GMT
View Forum Message <> Reply to Message

I can do like this:

```
struct MyPanel : Panel {

 virtual void Paint(Draw& w)override
 {
  w.DrawText(p.x, p.y, "#", Arial(30), Red);
 }

 virtual void Key((dword key, int count) override
 {
    switch (key)
  {
   case K_W:
    p.y-=1;
    break;
   case K_S:
    p.y+=1;
    break;
   case K_A:
```

```
    p.x-=1;
    break;
   case K_D:
    p.x+=1;
    break;
   default:
    ;
  }
            return true;
}
```

... but I suspect in U++ there is another way more direct than this.

---

Subject: Re: Focus problem
Posted by Lance on Sat, 18 Dec 2021 16:17:42 GMT
View Forum Message <> Reply to Message

Hi Silvan:

That might be what you need to do.

Unless the Ctrl (in your case, ImageCtrl) has an overrided Key which allow you to assign a WhenKey callback etc where you can do a per-instance tailoring.

A Event/Gate is a member variable and is pretty expensive if U++ just randomly provide one for all possible events(like the WhenSuspiciousKey I defined for the MainWindow class). So not every Ctrl defined an WhenKey event. There might be something one can do on the libary level to make it both convenient and not unnecessarily expensive on memory cost, but for now, what you have done is right. If not sure, Ctrl-Left click into ImageCtrl class definition to verify there is no member event like WhenKey, or see it has the Key virtual function overrided at all.

---

Subject: Re: Focus problem
Posted by Lance on Sat, 18 Dec 2021 16:24:07 GMT
View Forum Message <> Reply to Message

for every virtual function(or even non-virtual ones), you can potentially define one or more Event/Gate, etc, to suit your needs. But remember there is cost associated with that. You need to balance between convenience and cost (in memory usage).

---

Subject: Re: Focus problem

Posted by Silvan on Sat, 18 Dec 2021 17:32:54 GMT
View Forum Message <> Reply to Message

Hi Lance,
I understand the cost, but my code above does not compile.
How can I specify a code for an event or a method for a Ctrl?
I'm a C low level programmer and I really like U++ and sometimes I use it
for front end gui stuff.
I suppose there should be some direct way to define the code for event and method?
Similar to C#, ore VCL.
Or that is not possible for permormance reason?

Thank you

PS: I clarify better my doubt.
There are plenty of example that show how simple and fast is to program the U++ GUI framework by easily define your handler to event and method.
But they always refers to TopWindows or menu or statusbar (using overrid) , I can't find an example for other gadget like ImageCtrl.
Maybe I'm confused by the sentence that told THISBACK is deprecated, maybe that is the simplest way. But I'm now confused.

---

Subject: Re: Focus problem
Posted by Silvan on Sun, 19 Dec 2021 11:23:17 GMT
View Forum Message <> Reply to Message

Here is a working code as intended:
some thought:
1) I used a new class derived from ImageCtrl, I suppose that is a little bit overkill for a simply overload of two method/event;
2) I had to use a global variable (Point p) otherwise not accessible from the new class Mypanel. Really bad.
3) I had to use the override Close method otherwise the program terminate with an error.

I suppose that U++ allow to do all this much better... I'm searching how hoping in some help!
Thank you
Silvan

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

using namespace Upp;
```

```
 Point p;

struct MyPanel : ImageCtrl {

  virtual void Paint(Draw& w)override
  {
   w.DrawRect(GetSize(), White());
   w.DrawText(p.x, p.y, "#", Arial(30), Red);
   Refresh();
  }

  virtual bool Key(dword key, int count) override
  {
   switch (key)
   {
    case K_W:
     p.y-=1;
     break;
    case K_S:
     p.y+=1;
     break;
    case K_A:
     p.x-=1;
     break;
    case K_D:
     p.x+=1;
     break;
    default:
     ;
   }
   Refresh();
        return true;
  }

  void LeftDown(Point p, dword keyflags) override
  {
   SetFocus();
  }

};

struct MainWindow : TopWindow {


  EditString inputtext;
  MyPanel panel;

  void Close() override
```

```
  {
   delete this;
  }

 // Costruttore dove inserisci le inizializzazioni
  MainWindow()
  {

   Title("Test Focus").Zoomable().Sizeable();
   Add(inputtext.TopPosZ(0, 16).HSizePos());
   Add(panel.VSizePos(26, 0).HSizePos(0, 0));

   inputtext <<= "test";

   SetRect(0, 0, 300, 300);
   p.x = 150;
   p.y = 150;
  }
};

GUI_APP_MAIN
{
(new MainWindow)->OpenMain();
Ctrl::EventLoop();
}
```

---

Subject: Re: Focus problem
Posted by Lance on Sun, 19 Dec 2021 13:22:44 GMT
View Forum Message <> Reply to Message

Hi Silvan:

The way I define WhenSuspiciousKey is a typical way to define an Event. First, the Ctrl or its direvative need to provide a chance for the event in a (relevant) overrided virtual function. In my case, I overrided

bool Key(...)

In my MainWindow::Key(...)
I allowed WhenSuspiciousKey be called at certain time. What is WhenSuspiciousKey? I define it as a member variable of MainWindow of type Event<>

By add this member variable to MainWindow, the user of MainWindow can assign a callback to an instance(object) of this type by

MainWindow m;

m<<[]{...};

There is no magic here. C have simlar (but much limited ) mechanism. WhenSuspiciousKey is like a function pointer to which you can assign a callback. In order for this callback to actually be called, some where, you need to check and call it(in our case, it's in the MainWindow::Key(...) virtual function).

Now, Key is a virtual function, you cannot assign a callback to it. On the other hand, an object of MainWindow, m, has a public member variable name WhenSuspciciousKey, which is of type Event<>. You can access this member variable. One way is by something like m.WhenSuspiciousKey<<[]{};

---
PS: On a second thought, use C function pointer as example:

```
void (*WhenWhat)(int);

void func1(int p)
{
  if(WhenWhat)
    WhenWhat(p);
}

void callback(int p)
{
}

WhenWhat=callback; // this is good
// just like
// WhenSuspiciousKey<<[]{};
// is fine
func1=callback; // this won't compile
// just like
// Key<<[]{};
// won't compile.
```

---

Subject: Re: Focus problem
Posted by Lance on Sun, 19 Dec 2021 13:31:40 GMT
View Forum Message <> Reply to Message

1): Derive MyPanel from ImageCtrl to supply customized Paint and Key is the right thing to do, at the moment at least, unfortunately.

3): Overriding MyPanel::Close to delete this is unnecessary and considered bad practice (in this

case at least).

By overriding MyPanel::Close in your way, you basically enforce that MyPanel object have be be allocated from heap. U++ doesn't prohibit you from allocating Ctrl derivatives from heap, but it's much more often we have its object contained and be allocated from stack(faster and less memory usage). Or it can even be global, static etc.

In your case, if you really want your object be allocated from heap, you can use U++ provided smart pointer One to manage it for you. See example in the next reply.

---

Subject: Re: Focus problem
Posted by Lance on Sun, 19 Dec 2021 13:44:42 GMT
View Forum Message <> Reply to Message

2): put
Point p;
as global is unnecessary and highly likely conceptually wrong: what if you want multiple but independent windows who need to maintain their own current positions?.

The following revised code takes care of that, also demonstrate a simple use of One.

U++ is created by powerful programmers who actually use it. There are a lot of convenience utilities like One,DUMP,LOG, etc (I don't know many either, but read U++ source code, read old posts, etc and build up this knowledge; it will significantly increase your productivity: almost in all situation you need a specific tool or facility, someone before you has encounter it and figure out a smart solution).

BTW, welcome to U++ community!

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

using namespace Upp;

struct MyPanel : ImageCtrl {
 Point p;

 virtual void Paint(Draw& w)override
 {
  w.DrawRect(GetSize(), White());
  w.DrawText(p.x, p.y, "#", Arial(30), Red);
  Refresh();
 }

 virtual bool Key(dword key, int count) override
 {
  switch (key)
  {
```

```
   case K_W:
    p.y-=1;
    break;
   case K_S:
    p.y+=1;
    break;
   case K_A:
    p.x-=1;
    break;
   case K_D:
    p.x+=1;
    break;
   default:
    ;
  }
  Refresh();
       return true;
 }

 void LeftDown(Point p, dword keyflags) override
 {
  SetFocus();
 }

};

struct MainWindow : TopWindow {


  EditString inputtext;
  MyPanel panel;

//  void Close() override
//  {
//   delete this;
//  }

 // Costruttore dove inserisci le inizializzazioni
  MainWindow()
  {

   Title("Test Focus").Zoomable().Sizeable();
   Add(inputtext.TopPosZ(0, 16).HSizePos());
   Add(panel.VSizePos(26, 0).HSizePos(0, 0));

   inputtext <<= "test";

   SetRect(0, 0, 300, 300);
```

```
   panel.p.x = 150;
   panel.p.y = 150;
  }
};

GUI_APP_MAIN
{
	// dynamically allocated MainWindow
 One<MainWindow> m;
 m.Create<MainWindow>().OpenMain();

	// allocated from Stack.
	MainWindow().OpenMain();

	// because now Point p is per instance (not global)
	// you can control each independent of the other.
//(new MainWindow)->OpenMain();
 Ctrl::EventLoop();
}
```

---

Subject: Re: Focus problem
Posted by Silvan on Sun, 19 Dec 2021 14:37:54 GMT
View Forum Message <> Reply to Message

Lance wrote on Sun, 19 December 2021 14:311): Derive MyPanel from ImageCtrl to supply customized Paint and Key is the right thing to do, at the moment at least, unfortunately.

3): Overriding MyPanel::Close to delete this is unnecessary and considered bad practice (in this case at least).

By overriding MyPanel::Close in your way, you basically enforce that MyPanel object have be be allocated from heap. U++ doesn't prohibit you from allocating Ctrl derivatives from heap, but it's much more often we have its object contained and be allocated from stack(faster and less memory usage). Or it can even be global, static etc.

In your case, if you really want your object be allocated from heap, you can use U++ provided smart pointer One to manage it for you. See example in the next reply.

If I comment out MainWindows::Close (not MyPanel) than when I exit the program there is an exception at address .....
That appens with:

```
 (new MainWindow)->OpenMain();
 Ctrl::EventLoop();
```

With:

```
MainWindow m;
m.Run();
```

I can comment out Close whitout error on exiting.

---

Subject: Re: Focus problem
Posted by Silvan on Sun, 19 Dec 2021 14:59:41 GMT
View Forum Message <> Reply to Message

Lance wrote on Sun, 19 December 2021 14:311): Derive MyPanel from ImageCtrl to supply customized Paint and Key is the right thing to do, at the moment at least, unfortunately.

This is a big sentence. Are you saying that I could create a member fuction of MainWindos and call it throut a method of MainWindos
(like you did with WhenSuspiciousKey invoked inside the Key event of MainWindos) but I cannot attack directly WhenSuspiciousKey
to the Key event of ImageCtrl?

---

Subject: Re: Focus problem
Posted by Lance on Sun, 19 Dec 2021 15:16:49 GMT
View Forum Message <> Reply to Message

The memory leak thing you encountered can be either take care of by your way of override Close() or by the suggested way of using smart pointer One. The latter is preferred.

Key is a virtual function, it can create an WhenKey event (just like I created the WhenSuspiciousKey event). If a WhenKey event is created, either by yourself or by the libary you are using, you can assign a per-instance callback to tailor the particular object's behaviour. The virtual function Key is per-class. And it's not assignable. It's constant.

Please read carefully my previous 3-4 replies. All I can tell are there.

---

Subject: Re: Focus problem
Posted by Lance on Sun, 19 Dec 2021 15:24:26 GMT

View Forum Message <> Reply to Message

Quote:
1): Derive MyPanel from ImageCtrl to supply customized Paint and Key is the right thing to do, at the moment at least, unfortunately.

By this sentence, I mean what you have done with MyPanel is correct and necessary. I am also hinting that, what you are expecting, to be able to tailor an object's behavior by supplying some callbacks instead of having to define new class for something quite trivial, is reasonable. Unfortunately due to cost concerns etc., is currently unavailable, except for some Ctrl derivatives who indeed supply one.

Subject: Re: Focus problem
Posted by Silvan on Sun, 19 Dec 2021 18:31:45 GMT
View Forum Message <> Reply to Message

Lance wrote on Sun, 19 December 2021 16:16
Please read carefully my previous 3-4 replies. All I can tell are there.

I have a lot to study... thank you!

Subject: Re: Focus problem
Posted by Lance on Mon, 20 Dec 2021 12:43:58 GMT
View Forum Message <> Reply to Message

So do I. U++ is vast. You are welcome!