
Subject: Deleting Ctrl from within itself

Posted by [ultimatecoding](#) on Tue, 16 Jul 2019 03:45:34 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello,

What is the correct way to make a control derived from Ctrl self-destruct? For example, when we press a button, we want that particular control to disappear and free its resources. The following code works from the parent control level but crashes the program when triggered by a child control.

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

```
using namespace Upp;
```

```
struct ClosableCtrl : Ctrl
```

```
{
    Button closeButton;
    int index;

    Event<> WhenClosePressed;

    ClosableCtrl(int idx) :
        index(idx)
    {
        Add(closeButton.SetLabel(AsString(idx)).HSizePos(10, 10).VSizePos(5, 5));
        closeButton.WhenPush = [=] { OnClosePressed(); };
    }
}
```

```
~ClosableCtrl()
{
    LOG("ClosableCtrl destroyed");
}
```

```
virtual void Paint(Upp::Draw& w) override
{
    w.DrawRect(GetSize(), Green());
}
```

```
int GetIndex() const
{
    return index;
}
```

```
void SetIndex(int idx)
{
    index = idx;
}
```

```

void OnClosePressed()
{
    WhenClosePressed();
}

};

struct CtrlPanel : Ctrl
{
    Vector<ClosableCtrl*> closableCtrls;

    CtrlPanel(int n)
    {
        for (int i = 0; i < n; ++i) {
            ClosableCtrl *cc = new ClosableCtrl(i);
            cc->WhenClosePressed = [=] { RemoveCtrl(cc->GetIndex()); };
            Add(cc->HSizePos(0, 0).TopPos(i * 100, 100));
            closableCtrls << cc;
        }
    }

    ~CtrlPanel()
    {
        for (ClosableCtrl *cc : closableCtrls) {
            delete cc;
        }
    }

    virtual void Paint(Upp::Draw& w) override
    {
        w.DrawRect(GetSize(), White());
    }

    void RemoveCtrl(int idx)
    {
        delete closableCtrls[idx];
        closableCtrls.Remove(idx);
        for (int i = 0, n = int(closableCtrls.size()); i < n; ++i) {
            closableCtrls[i]->SetIndex(i);
        }
    }
};

struct MainWindow : TopWindow
{
    CtrlPanel ctrlPanel;

```

```

MainWindow() :
    ctrlPanel(5)
{
    CenterScreen().Sizeable().Zoomable();
    Title("Event Test");

    Add(ctrlPanel.HSizePos(5, 5).VSizePos(5, 5));
    ctrlPanel.RemoveCtrl(1); // This works
}
};

GUI_APP_MAIN
{
    MainWindow mainWin;
    mainWin.Run();
}

```

If I press one of the buttons, a read access violation error is thrown at
Ctrl *Ctrl::GetTopCtrl()

```

{
    GuiLock __;
    Ctrl *q = this;
    while(q->parent)
        q = q->parent;
    return q;
}

```

Also, what is the best way to store multiple controls dynamically in cases like this? According to my understanding of [https://www.ultimatepp.org/www\\$uppweb\\$overview_en-us.html](https://www.ultimatepp.org/www$uppweb$overview_en-us.html), it is not required that an element have a copy constructor, but when I try things like Vector<ClosableCtrl> or Array<ClosableCtrl>, I keep getting errors related to copy constructors.

Thank you.

Subject: Re: Deleting Ctrl from within itself
 Posted by [mirek](#) on Fri, 19 Jul 2019 13:27:04 GMT
[View Forum Message](#) <> [Reply to Message](#)

- use Array
- key trick to situations like this is to use PostCallback to get "on top" of stack

```

#include <CtrlLib/CtrlLib.h>

using namespace Upp;

struct MyApp : TopWindow {

```

```
Array<Ctrl> ctrl;
```

```
MyApp() {  
    auto& b = ctrl.Create<Button>();  
    Add(b.TopPos(10).LeftPos(10, 100));  
    b.SetLabel("Close me!");  
    b << [&] {  
        PostCallback([=] { ctrl.Remove(0); });  
    };  
}  
};
```

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