Subject: Random problems with destructors in GUI classes
Posted by koldo on Thu, 08 Feb 2018 09:11:35 GMT
View Forum Message <> Reply to Message

Hello all

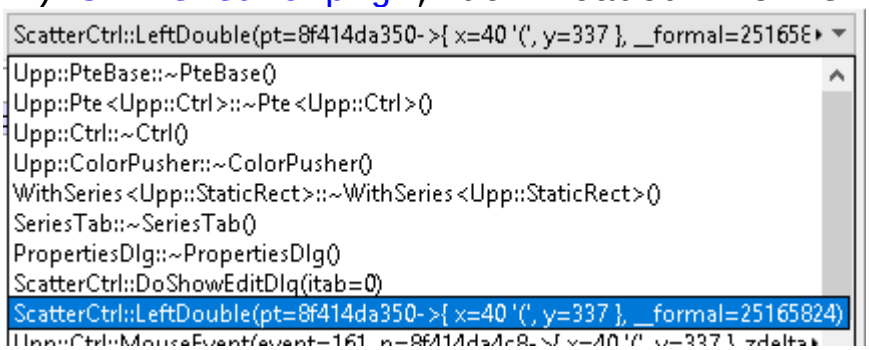
I have random problems with GUI controls. Sometimes with DEBUG or RELEASE, when a control is destructed an EXCEPTION_ACCESS_VIOLATION exception raises. This happens in ScatterCtrl.
For example, function ScatterCtrl::DoShowEditDlg() calls PropertiesDlg(*this, itab).Run(true) opening a dialog.
Sometimes, when this dialog is closed, this errror raises. The function call stack represets basically destructors:

It is like there is a problem in the destructors order, so that in this case, when ColorPusher destructor is called, parent controls has been already destructed.

I have seen that many or maybe all GUI controls have virtual destructors. Maybe it has no sense but, should GUI user controls destructors have to be also virtual?

File Attachments

1) Sin título.png , downloaded 1454 times



Subject: Re: Random problems with destructors in GUI classes
Posted by dolik.rce on Thu, 08 Feb 2018 10:12:59 GMT
View Forum Message <> Reply to Message

Hi Koldo,

You are right about the virtual destructors. If you have virtual inheritance, you must always declare virtual destructor, even if it doesn't do anything. Have at the discussion here for some details why it is required.

Best regards,
Honza

Subject: Re: Random problems with destructors in GUI classes
Posted by koldo on Thu, 08 Feb 2018 10:25:50 GMT
View Forum Message <> Reply to Message

Thank you Honza

This would mean that all GUI user classes must have virtual destructors?

Subject: Re: Random problems with destructors in GUI classes
Posted by dolik.rce on Thu, 08 Feb 2018 11:40:05 GMT
View Forum Message <> Reply to Message

koldo wrote on Thu, 08 February 2018 11:25Thank you Honza

This would mean that all GUI user classes must have virtual destructors?
Yes, because they are probably always released via Ctrl* pointer. Non-virtual destructor is not called in such case.

Honza

Subject: Re: Random problems with destructors in GUI classes
Posted by koldo on Thu, 08 Feb 2018 12:02:00 GMT
View Forum Message <> Reply to Message

Oupss! Thank you again

I should have to review many classes to include this.

Probably I have not realised about that, but we would have to include this in the documentation.

Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Fri, 09 Feb 2018 12:36:36 GMT
View Forum Message <> Reply to Message

koldo wrote on Thu, 08 February 2018 11:25Thank you Honza

This would mean that all GUI user classes must have virtual destructors?

'virtuality' of destructor is inherited. Since Ctrl has virtual destructor, it is not really necessarry to declare destructor virtual in derived classes.

Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Fri, 09 Feb 2018 12:48:29 GMT
View Forum Message <> Reply to Message

koldo wrote on Thu, 08 February 2018 10:11Hello all

I have random problems with GUI controls. Sometimes with DEBUG or RELEASE, when a control is destructed an EXCEPTION_ACCESS_VIOLATION exception raises. This happens in ScatterCtrl.
For example, function ScatterCtrl::DoShowEditDlg() calls PropertiesDlg(*this, itab).Run(true) opening a dialog.
Sometimes, when this dialog is closed, this errror raises. The function call stack represets basically destructors:

It is like there is a problem in the destructors order, so that in this case, when ColorPusher destructor is called, parent controls has been already destructed.

I have seen that many or maybe all GUI controls have virtual destructors. Maybe it has no sense but, should GUI user controls destructors have to be also virtual?

With very little info you have provided I would actually think that the problem here is properties dialog is already deleted when the constructor is invoked - but that is just the first thing I would check.

I have tried to identify the problem in trunk, but all seems OK there. Is this trunk code or some new development?

Subject: Re: Random problems with destructors in GUI classes
Posted by koldo on Sat, 10 Feb 2018 22:53:48 GMT
View Forum Message <> Reply to Message

Hello Mirek

I did not know that 'virtuality' of destructor is inherited. In fact high level U++ GUI classes like ArrayCtrl have empty virtual destructors.

Now all code is uploaded.

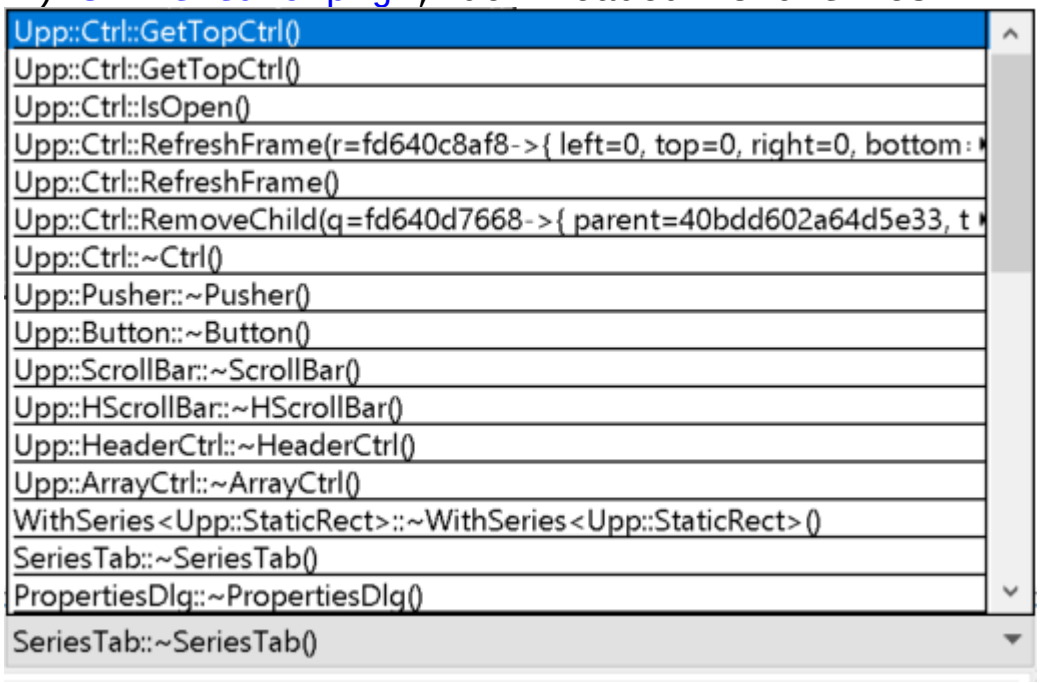
Virtualizing all destructors has not solved the issue.

It is like some destructor supposes that parent control already has not been closed.

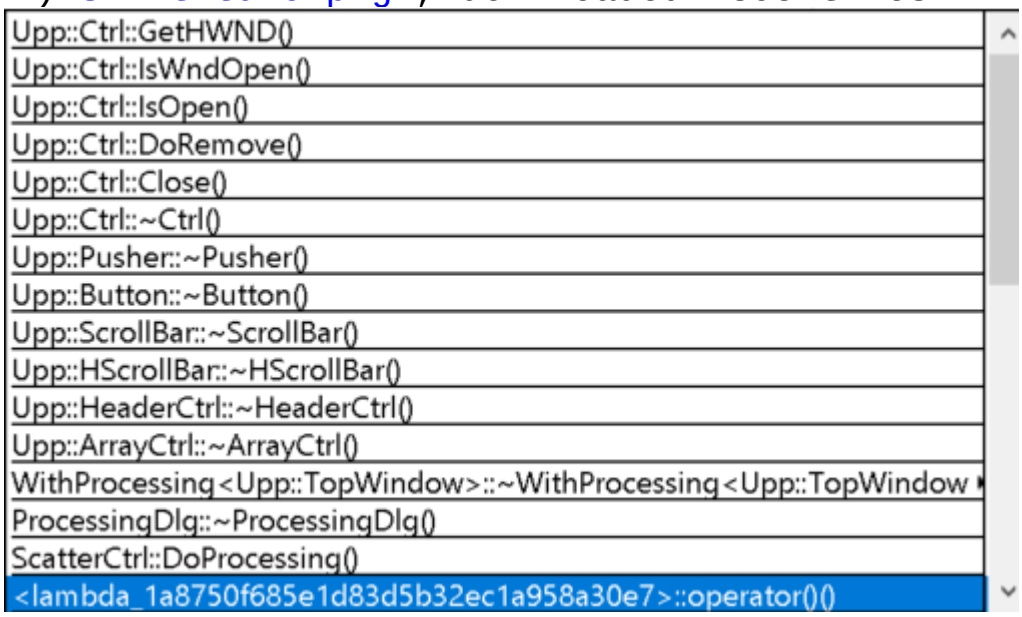
Two samples:

File Attachments

1) Sin título.png , downloaded 1310 times



2) Sin título.png , downloaded 1395 times



Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Sun, 11 Feb 2018 07:25:03 GMT
View Forum Message <> Reply to Message

koldo wrote on Sat, 10 February 2018 23:53Hello Mirek

I did not know that 'virtuality' of destructor is inherited. In fact high level U++ GUI classes like

ArrayCtrl have empty virtual destructors.

That was workaround for some old compiler bug.

Quote:
It is like some destructor supposes that parent control already has not been closed.

Indeed. However, I am unable to reproduce so far.

Is this some sort of new behaviour? (Like: has this started to happen since some time? Have you changed anything in ScatterCtrl?)

Any tips how to reproduce?

Quote:
Two samples:

[/quote]

BTW, you can copy backtrace to clipboard using Debug/Copy backtrace

---

Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Sun, 11 Feb 2018 07:33:34 GMT
View Forum Message <> Reply to Message

```
void PropertiesDlg::OnClose()
{
 measures.Change();

 RejectBreak(IDOK);
 Close(); Close();
}
```

It should not be a reason for these problems, but Break should be enough here. Why do you call Close twice after that?

Mirek

Subject: Re: Random problems with destructors in GUI classes

Posted by koldo on Sun, 11 Feb 2018 08:27:23 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 11 February 2018 08:33
void PropertiesDlg::OnClose()
{
 measures.Change();

 RejectBreak(IDOK);
 Close(); Close();
}

It should not be a reason for these problems, but Break should be enough here. Why do you call Close twice after that?

Mirek
It was necessary in the past. Now is not. I have just deleted it.

Subject: Re: Random problems with destructors in GUI classes
Posted by koldo on Mon, 12 Feb 2018 21:24:57 GMT
View Forum Message <> Reply to Message

Well, this does not solve the problem but solves the exception:

void ScatterCtrl::DoShowData()
{
 static DataDlg dlg;
 ONCELOCK {
  dlg.Init(*this);
 }
 dlg.Run(true);
}
It has been applied to all problematic dialogs. The problems have not been repeated.

Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Mon, 12 Feb 2018 22:06:48 GMT
View Forum Message <> Reply to Message

koldo wrote on Mon, 12 February 2018 22:24Well, this does not solve the problem but solves the exception:

void ScatterCtrl::DoShowData()
{
 static DataDlg dlg;

```
 ONCELOCK {
  dlg.Init(*this);
 }
 dlg.Run(true);
}
```
It has been applied to all problematic dialogs. The problems have not been repeated.

OK, that would mean the cause probably really is some destruction order issue. Now we should find and fix it for real...

BTW, why do you use appmodal mode (Run(true))? That is not standard.

Subject: Re: Random problems with destructors in GUI classes
Posted by koldo on Tue, 13 Feb 2018 09:53:40 GMT
View Forum Message <> Reply to Message

Quote:BTW, why do you use appmodal mode (Run(true))? That is not standard.

What do you recommend?

Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Sat, 17 Feb 2018 11:43:47 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 13 February 2018 10:53Quote:BTW, why do you use appmodal mode (Run(true))? That is not standard.

What do you recommend?

true blocks all application windows, which sometimes is not quite right. Just use Run(), which defaults to false and blocks only current main window.

Subject: Re: Random problems with destructors in GUI classes
Posted by koldo on Sun, 18 Feb 2018 09:06:16 GMT
View Forum Message <> Reply to Message

OK. Thank you.

Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Sat, 24 Feb 2018 09:28:02 GMT
View Forum Message <> Reply to Message

koldo wrote on Mon, 12 February 2018 22:24Well, this does not solve the problem but solves the exception:

```
void ScatterCtrl::DoShowData()
{
 static DataDlg dlg;
 ONCELOCK {
  dlg.Init(*this);
 }
 dlg.Run(true);
}
```

It has been applied to all problematic dialogs. The problems have not been repeated.

This is fatal error.

a) it will crash if you close scatter ctrl and create another one (because dlg will keep pointer to nonexisting ScatterCtrl)
b) it will behave incorrectly with 2 ScatterCtrls
c) it must work without such trick

Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Sat, 24 Feb 2018 11:40:44 GMT
View Forum Message <> Reply to Message

I have done a light cleanup of ScatterCtrl code, removing unnecessarry things. Might have fixed the issue, but unlikely.

In any case, I really have problem to reproduce the issue. Need your help.

Subject: Re: Random problems with destructors in GUI classes
Posted by koldo on Fri, 02 Mar 2018 14:00:48 GMT
View Forum Message <> Reply to Message

Finally the cause was totally unrelated with ScatterCtrl and was caused by SetTimeCallbacks not handled properly.

It has been difficult to find this, but the problems must not be swept under the carpet, and the root cause of them has to be found. A patch is not a solution

---

Subject: Re: Random problems with destructors in GUI classes
Posted by mirek on Fri, 02 Mar 2018 14:07:28 GMT
View Forum Message <> Reply to Message

koldo wrote on Fri, 02 March 2018 15:00Finally the cause was totally unrelated with ScatterCtrl

and was caused by SetTimeCallbacks not handled properly.

It has been difficult to find this, but the problems must not be swept under the carpet, and the root cause of them has to be found. A patch is not a solution

Excellent! Thank you for the hard work.

I know how complicated it is sometimes. I remember hunting RichEdit bug for two years - it was randomly crashing in my app, like once a month.... (it turned out to be invalid READ - most of time it harmlessly read some random data, but once a month it hit not yet reserved memory).
