Subject: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by alendar on Tue, 07 Sep 2010 03:07:51 GMT
View Forum Message <> Reply to Message

Hi,

I finally tracked down a bug that has been enbedded in my code since I converted from Windows 7 to Ubuntu 9.10, now Ubuntu 10.04 LTS 64-bit.

The error is intermittent but easy for me to reproduce.

In debug mode, from the log file.
Assertion failed in /home/jeff/upp/uppsrc/CtrlCore/X11Wnd.cpp, line 473
!IsChild() && !IsOpen()

Resource temporarily unavailable

A blank white popup label box remains on the screen, no text, until the program is terminated.

The code is from CtrlCore::X11Wnd.cpp::Ctrl::Create0(Ctrl *owner, bool redirect, bool savebits)

At the assertion around line 470.
ASSERT(!IsChild() && !IsOpen());

I determined that the IsOpen() test is failing.

I was able to fix it, but not sure if its the greatest solution.

The steps to reproduce:
1) Create multithreaded app. An ArrayCtrl is built in the main thread with 14 columns. The number of columns may load the system enough to cause it. A thread is spawned to Add rows and populate in background. GuiLock __ is used.
2) In main, call PopUpEx() to enable popup info.
3) Load enough data to enable scrolling.
4) Begin scrolling rapidly down with the mouse wheel and hovering over cells. Sometimes shifting between columns causes the failure to happen sooner.

Using LLOG, I determined its during the Add() call to ArrayCtrl.

Here's where the popup creation happens successfully:
POPUP: N3Upp12DisplayPopupE : 0x7fffeb7377c0(window 0x0)
Create N3Upp12DisplayPopupE : 0x7fffeb7377c0(window 0x0) [1045, 384] - [1087, 403] : (42, 19)

Note that the POPUP: statement precedes the create command.

Here's where the failure happens:
Invalidate N3Upp12DisplayPopupE : 0x7fffeb7377c0(window 0x66014d5) [0, 0] - [152, 19] : (152,

19)
 Create N3Upp12DisplayPopupE : 0x7fffeb7377c0(window 0x66014d5) [667, 579] - [819, 598] : (152, 19)

The window exists already since it is invalidated, and the the Create statement crashes because it already exists.

My solution was to add a Close() call if it was Open:

```
void Ctrl::Create0(Ctrl *owner, bool redirect, bool savebits)
{
 GuiLock __;
 ASSERT(IsMainThread());
 LLOG("Create " << Name() << " " << GetRect());
 if (IsChild())
     LLOG("Ctrl::Create0 IsChild = True");
 if (IsOpen()) {
  LLOG("Ctrl::Create0 IsOpen = True");
  Close(); // HACK
 }
 ASSERT(!IsChild() && !IsOpen());
```

Ubuntu: 2.6.32-24-generic #42-Ubuntu SMP Fri Aug 20 14:21:58 UTC 2010 x86_64 GNU/Linux
CPU: AMD Athlon(tm) 64 X2 Dual Core Processor 4600+
U++: 2667-lucid-amd64-nogtk (64 bit)
gcc: (Ubuntu 4.4.3-4ubuntu5) 4.4.3

I will try to dummy up a sample. Hopefully I can get a fix applied to the code since I get automatic upgrades.

Note: Fails on GTK version, too.

Subject: Re: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by alendar on Tue, 12 Oct 2010 01:33:43 GMT
View Forum Message <> Reply to Message

Wow, new record. O responses. I thought since it was a real bug it would get noticed. I have to use U++; there's no competitor I can find that is open source and actively maintained any better, but I'm disappointed.

Subject: Re: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by koldo on Wed, 13 Oct 2010 06:57:16 GMT

View Forum Message <> Reply to Message

Hello Alendar

Could you include a simple sample package?

Subject: Re: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by mirek on Sat, 16 Oct 2010 10:54:58 GMT
View Forum Message <> Reply to Message

alendar wrote on Mon, 11 October 2010 21:33Wow, new record. O responses. I thought since it was a real bug it would get noticed. I have to use U++; there's no competitor I can find that is open source and actively maintained any better, but I'm disappointed.

Sorry, been busy lately.

Might I as you to provide a test-case? Small package and instructions to crash it

Mirek

Subject: Re: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by bushman on Fri, 05 Jul 2013 02:05:05 GMT
View Forum Message <> Reply to Message

I've been experiencing basically a very similar problem in Windows 7 lately:
I'm running a MT app that spawns a background thread to populate an ArrayCtrl, while keeping the GUI available for user inputs. If I hover the mouse quickly over the ArrayCtrl while it is getting fresh rows, it's not really difficult to reproduce the same ASSERT(!IsChild() && !IsOpen()); crash around line 568 of Win32Win.cpp. I'm GuiLock __ -ing ArrayCtrl when adding rows.
I tried this hack, as Alendar suggested in his post, and had no more crashes since then:

```
void Ctrl::Create0(Ctrl::CreateBox *cr)
{
 GuiLock __;
 ASSERT(IsMainThread());
 LLOG("Ctrl::Create(parent = " << (void *)parent << ") in " <<UPP::Name(this) << LOG_BEGIN);
 if (IsOpen()) {
  LLOG("Ctrl::Create0 IsOpen = True");
  Close(); // HACK
 }
 ASSERT(!IsChild() && !IsOpen());
        ...
```

Can someone give a hint on what is possibly going on here?
tks!

kropniczki wrote on Thu, 04 July 2013 22:05I've been experiencing basically a very similar problem in Windows 7 lately:
I'm running a MT app that spawns a background thread to populate an ArrayCtrl, while keeping the GUI available for user inputs. If I hover the mouse quickly over the ArrayCtrl while it is getting fresh rows, it's not really difficult to reproduce the same ASSERT(!IsChild() && !IsOpen()); crash around line 568 of Win32Win.cpp. I'm GuiLock __ -ing ArrayCtrl when adding rows.
I tried this hack, as Alendar suggested in his post, and had no more crashes since then:

```
void Ctrl::Create0(Ctrl::CreateBox *cr)
{
 GuiLock __;
 ASSERT(IsMainThread());
 LLOG("Ctrl::Create(parent = " << (void *)parent << ") in " <<UPP::Name(this) << LOG_BEGIN);
 if (IsOpen()) {
  LLOG("Ctrl::Create0 IsOpen = True");
  Close(); // HACK
 }
 ASSERT(!IsChild() && !IsOpen());
       ...
```

Can someone give a hint on what is possibly going on here?
tks!

Well, there seems to still be a problem of MT GuiLock/Call design.

The root of problem is stupid M$ decision that binds windows and event loops to threads. That makes practically only possible to create windows and run event loops in the main thread.

We have tried to workaround this by "Ctrl::Call", which is somewhat supposed to "call" function in the main thread. The problem is that in order to do that, Call has to unlock GuiLock so that the main thread has the chance to perform the request.

So the culprit of this crash is that non-main thread does something in ArrayCtrl, which in turn invokes void DisplayPopup::Sync() and there is innocent looking code like:

```
if(!IsOpen())
Ctrl::PopUp(ctrl, true, false, false);
```

Anyway, what really happens is that Ctrl::PopUp in non-main thread has to use Call to get into main thread, but as main thread is invoked, it opens the window on its own (in certain situations anyway), which leads to the crash...

Now I am sort of undecided about what course to take to fix this. I am afraid it is quite incorrect that GuiLock is "broken in" by the main thread, but I am afraid that trying to fix that with some

more locks or something would just get us into more and more complicated model, with some similar hard to detect bugs.

Alternatively, I am starting to thing that perhaps easiest is to ban creation of windows in non-main thread (Prompts could perhaps be supported as exception, DisplayPopup::Sync would have to be rewritten).

Subject: Re: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by mirek on Sat, 06 Jul 2013 09:37:02 GMT
View Forum Message <> Reply to Message

Should be fixed (but the issue remains).

Subject: Re: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by bushman on Sat, 06 Jul 2013 15:39:33 GMT
View Forum Message <> Reply to Message

Tks for the prompt response.

Quote:The root of problem is stupid M$ decision that binds windows and event loops to threads. That makes practically only possible to create windows and run event loops in the main thread.

I thought it was something rather OS-independent, since the issue apparently occurs under Linux too, according to Alendar's posting in this topic. Anyway, I wonder, would it be worth to ask how M$ does it? In other words, how a pure M$ app is supposed to be developed so that it does what I want without crashing? I mean, is there some M$ recipe for creating new windows not in main thread and still be able to run event loops in it without crashing? I don't know, maybe such an investigation could bring up new insights on which course to take to fix the problem, for I guess purely M$-based progs must end up having to do it one way or another.

Quote:Alternatively, I am starting to thing that perhaps easiest is to ban creation of windows in non-main thread (Prompts could perhaps be supported as exception, DisplayPopup::Sync would have to be rewritten).

If you decide to do so, does it mean we will no longer be able do stuff like adding Ctrls in a background thread dynamically, while 'simultaneously' enabling user interaction with GUI, or would new DisplayPopup::Sync code otherwise still make it possible?

tks.

Subject: Re: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by mirek on Sat, 06 Jul 2013 18:54:53 GMT

kropniczki wrote on Sat, 06 July 2013 11:39Tks for the prompt response.

Quote:The root of problem is stupid M$ decision that binds windows and event loops to threads. That makes practically only possible to create windows and run event loops in the main thread.

I thought it was something rather OS-independent, since the issue apparently occurs under Linux too, according to Alendar's posting in this topic.

Well, the decision then requires all that main-thread GUI stuff, so other platforms have to follow if we want to be crossplatform compatible...

Quote:
Anyway, I wonder, would it be worth to ask how M$ does it? In other words, how a pure M$ app is supposed to be developed so that it does what I want without crashing? I mean, is there some M$ recipe for creating new windows not in main thread and still be able to run event loops in it without crashing? I don't know, maybe such an investigation could bring up new insights on which course to take to fix the problem, for I guess purely M$-based progs must end up having to do it one way or another.

Well, I guess it is not really that much needed.... I mean, opening windows in non-main threads is not really that much useful.

Quote:
Quote:Alternatively, I am starting to thing that perhaps easiest is to ban creation of windows in non-main thread (Prompts could perhaps be supported as exception, DisplayPopup::Sync would have to be rewritten).

If you decide to do so, does it mean we will no longer be able do stuff like adding Ctrls in a background thread dynamically, while 'simultaneously' enabling user interaction with GUI, or would new DisplayPopup::Sync code otherwise still make it possible?

It will be OK. (Actually it IS OK, as changes are commit). Only thing you cannot do easily anymore is to directly open window from another thread or run message loop. But even for that you have still easy workaround, just use Call to have Callback performed on the main thread. You just have to consider that Call unlocks GuiLock...

You can check updated GuiLock example... (Even if actually it would work without change, as Prompts are "fixed" to run from non-main thread).

Mirek
Mirek