Subject: Re: ArrayCtrl: GPF when thread Add(), PopUpEx, and Scroll collide
Posted by mirek on Fri, 05 Jul 2013 17:54:28 GMT
View Forum Message <> Reply to Message

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).