Subject: Re: how to use timer id?
Posted by mrjt on Thu, 22 Jul 2010 08:58:14 GMT
View Forum Message <> Reply to Message

I just cannot understand what it is you are trying to do

You are correct that my previous example wasn't threadsafe. Here is a threadsafe version. As far as I can tell ALL timers will executed in the GUI thread.

```
#include "CtrlLib/CtrlLib.h"

using namespace Upp;

class MyWindow : public TopWindow
{
 typedef MyWindow CLASSNAME;
 int count;
public:
 MyWindow() : count(0)     {}
 virtual void Paint(Draw& w)   { TopWindow::Paint(w); w.DrawText(4, 4, AsString(count)); }
 void Count()       { ASSERT(Thread::IsMain()); ++count; Refresh(); }
};

void MyThread(Callback cb)
{
 TimeCallback timer;
 for (int i = 0; i < 50; ++i) {
  if (Thread::IsShutdownThreads())
   return;
  Sleep(100);
 }
 timer.Set(-1000, cb);
 for (int i = 0; i < 50; ++i) {
  if (Thread::IsShutdownThreads())
   return;
  Sleep(100);
 }
 timer.Kill();
}

GUI_APP_MAIN
{
 MyWindow wnd;
 Thread thrd;

 wnd.SetRect(RectC(0, 0, 200, 200));
 wnd.CenterScreen();

 thrd.Run(callback1(MyThread, callback(&wnd, &MyWindow::Count)));
```

```
 wnd.Run();

 Thread::ShutdownThreads();
}
```

Quote:The problem is, I do not know whether the timer is already envoked when I kill it
Of course. You can be sure that the timer has been created, but not whether it has been executed. If you need guaranteed excution than you should call the function directly (using GuiLock if it needs GUI access)

Quote:Plus, why did you test Thread::IsShutdownThreads()?
Because I call Thread::ShutDownThreads before closing. This ensures that all threads are finished, otherwise I get heap leaks from dangling threads. I check IsShutdownThreads so that I can terminate the thread prematurely, if you take them out you'd have to wait for the thread to finished naturally.
