Subject: Sometimes, you don't want to update the GUI...
Posted by tvanriper on Mon, 07 Jan 2008 18:42:02 GMT
View Forum Message <> Reply to Message

Sometimes, when you're working with very fast operations, you want to keep the user informed of progress, but you don't want to consume a lot of time telling the user about what's going on.

To this end, I created a simple little class to help me (I think this will also work with linux, provided you're using Core/Core.h):

```
class GuiWaitTest
{

protected:
 DWORD old;
 DWORD now;
 unsigned int delay;
 void init()
 {
  now = ::GetTickCount();
  old = now - delay;
 }

public:
 GuiWaitTest(void) : delay(100)
 {
  init();
 }

 GuiWaitTest( unsigned int delay ) : delay( 100 )
 {
  init();
 }

 GuiWaitTest& operator=( const unsigned int delay )
 {
  this->delay = delay;
  return *this;
 }

 bool operator()()
 {
  now = GetTickCount();
  bool result = ::abs(static_cast<long>( now - old ) ) >= static_cast<long>(delay);
  if ( result )
  {
   old = now;
```

```
  }
  return result;
 }

 ~GuiWaitTest(void)
 {
 }
};
```

So, you'd use this class like this:

```
for ( MyVar var = SomeOperation(); var.GetData().IsNotNull(); var = SomeOperation() )
{
  // Here, 'check_gui' is a member variable of type GuiWaitTest.
  // my_control is a read-only control in the GUI that only displays state.
  // as such, my_control is only displaying progress... it has no other purpose.
  if ( this->check_gui() )
  {
     my_control.SetData( var.GetData() );
     my_control.ProcessEvents();
  }
}
```

The class is designed to have other values as desired... in case 1/10th of a second is too fast or not fast enough for you (I use 1/10th as it's slow enough to prevent flicker, but fast enough to show that the system is working hard)... it works under the principle that the human eye can only detect a change after 1/10th of a second.

I'm tempted to create a class that turns a control into a 'sometimes displays info to user' style control, so you simply set the control's state and it either ignore you, or updates itself for the user to see, depending upon this class's test.

EDIT

I erred in the sample-usage code. This should look more correct.
