Subject: Re: The right way to use CoDo with GuiLock?
Posted by Oblivion on Sat, 11 Jun 2022 12:43:42 GMT
View Forum Message <> Reply to Message

Hello Mountacir,

There can be several approaches here, but I'd suggest these two:

```
void CoTest::CoForTest1()
{
 list.Clear();
 Vector<int> v;
 CoFor(100, [=, &v](int i) {
  CoWork::FinLock();
  v.Add(i);
 });
 std::for_each(v.begin(), v.end(), [=](auto n) { list.Add(n); });
}

void CoTest::CoForTest2()
{
 list.Clear();
 GuiUnlock __; // Unlock the global Gui Mutex so it can be acquired by the workers...
 CoFor(100, [=](int i) {
  GuiLock __; // Acquire ...
  list.Add(i);
 });
}
```

CoFor is a convenience function that does what you just do in your example: iteration.

Both CoDo and CoFor are blocking functions. They will wait for the workers to join.

So, first you will need to release the gui lock of main gui thread, using the GuiUnlock class.

IMO, however, CoForTest1() would be a better and safer way, as the CoForTest2 will start blocking the GUI when the item count is high (try with N=10000 and see the difference in the results).

P.s:

This is your example, done in the right way:

```
void CoTest::CoDoTest()
```

```
{
 list.Clear();
 std::atomic<int> ii(0);
 GuiUnlock __;
 CoDo([=, &ii] {
  GuiLock __;
  for(int i =ii++; i < 100 ; i=ii++) {
   list.Add(i);

  }
 });
}
```

Best regards,
Oblivion
