Subject: Re: Thoughts about alternative approach to multithreading
Posted by Mindtraveller on Fri, 14 Nov 2008 09:09:00 GMT
View Forum Message <> Reply to Message

OK, I`d like to inform on some progress on "alternative" multithreading. It looks like I finally managed to figure how to apply it to U++ correctly (even in previous example). I`ll describe it as soon as class is ready and tested.

For now I`ve made little investigation on callbacks efficiency as I`d like to use them in my implementation. Some simple test was made:

```
#include <Core/Core.h>
#include <math.h>

using namespace Upp;

class TestClass
{
public:
 void func(int a, int b)
 {
  int c = a + b*b;
  double xx = sin(a+b*1.34-3.14159252596428);
  double yy = cos((double) (b-a));
  x += ceil(xx*yy*(c-(b << 4)+(a*b)));
 }

private:
 int x;
};

CONSOLE_APP_MAIN
{
 TestClass tc;

 Callback2<int,int> cb = callback(&tc, &TestClass::func);
 dword cnt = 0;
 for (int j=0; j<50; ++j)
 {
  Cout() << FormatInt(j) << " ";
  dword t1 = GetTickCount();
  for (int i=0; i<2000000; ++i)
  {
   cb(100,500);
   //tc.func(100, 500);
  }
  cnt += GetTickCount()-t1;

  Sleep(1500);
 }
 Cout() << "\n" << FormatInt(cnt/j);
}
```

Averaged values were calculated for three cases:
1) Plain call
tc.func(100, 500); inside loop

2) Callback call
cb(100,500); inside loop

3) Callback stack variable creation+call+destruction
Callback2<int,int> cb = callback(&tc, &TestClass::func);
cb(100,500);
inside loop

On my PC I have these values:
1) 594
2) 592
3) 728

Conclusion: U++ callbacks are very efficient. Even for simple functions it takes very small tradeoff to use callbacks instead of plain member function calls. Creating and destroying of callback stack variable takes some time. Not very big though but it is better to avoid it.

Investigation continues.