Subject: I request the implementation of callback5
Posted by aftershock on Wed, 09 May 2018 19:23:27 GMT
View Forum Message <> Reply to Message

Hi,
I request the implementation of callback5.
Can you implement it?

Regards
A.

Subject: Re: I request the implementation of callback5
Posted by Oblivion on Wed, 09 May 2018 21:00:10 GMT
View Forum Message <> Reply to Message

Quote:Hi,
I request the implementation of callback5.
Can you implement it?

Regards
A.

Hello Aftershock,

Callback is depreceated. Is there any reason for not using Function, Event, or Gate (the latter two are actually aliases for Function templates with specific return types (void and bool)) that can take any number of arguments? They are compatible with Callback, and superior.

E.g.

```
Event<int, int, int, int, int>       event; // returns void;
Gate<int, int, int, int, int>        gate;  // returns bool;
Function<T(int, int, int, int, int)> func;  // returns type T;
```

Best regards,
Oblivion

Subject: Re: I request the implementation of callback5
Posted by aftershock on Thu, 10 May 2018 08:13:57 GMT
View Forum Message <> Reply to Message

I will use anything else if it works..

What do they do?
What is difference among them?
Advantages/disadvantages?
OK...I see the difference is in return type.

Subject: Re: I request the implementation of callback5
Posted by Oblivion on Thu, 10 May 2018 08:33:31 GMT
View Forum Message <> Reply to Message

Quote:
I will use anything else if it works..
What do they do?
What is difference among them?
Advantages/disadvantages?

They are basically the same. You can consider the Upp::Function (and its two derivatives Event, and Gate) as the re-implemantation of the old U++ Calllback mechanism, using the C++11 features such as variadic templates.

From the U++ official documentation:

Upp::Function is wrapper to represent callable operation. It is similar to std::function with two differences:

- Calling empty Function is allowed (and NOP). Returns zero.
- Functions can be combined (chained) using operator<<

So,

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

using namespace Upp;

struct Foo {
 Event<int, int> WhenAddition;
 void DoAddition(int a, int b) { WhenAddition(a, b); }
};

CONSOLE_APP_MAIN
{
```

```
 StdLogSetup(LOG_COUT);

 // I am explicitly specifying the type here for educational purpose.
 // Normally you can simply use auto, where it is proper.
 Event<int, int> Addition = [=](int a, int b) {
  LOG("Foo::DoAddition: " << a + b);
 };

 Gate<int, int> Subtraction = [=](int a, int b) {
  return a - b > 0;
 };

 Function<int(int, int)> Multiplication = [=](int a, int b) {
  return a * b;
 };

 Foo myfoo;
 myfoo.WhenAddition = pick(Addition);

 int a = 10, b = 5;

 myfoo.DoAddition(a, b);

 if(Subtraction(a, b))
  LOG("a is greater than b");

 LOG(Multiplication(a, b));
}
```

Best regards,
Oblivion

Subject: Re: I request the implementation of callback5
Posted by aftershock on Thu, 10 May 2018 16:53:14 GMT
View Forum Message <> Reply to Message

What about thisback?

E,g
threads[free_index].Run (THISBACK5 (execute_bot_in_background, bot1, a, pick(params), result_mode, stat_group_id));

Subject: Re: I request the implementation of callback5

Posted by Oblivion on Thu, 10 May 2018 17:15:52 GMT
View Forum Message <> Reply to Message

Quote:What about thisback?

E,g
threads[free_index].Run (THISBACK5 (execute_bot_in_background, bot1, a, pick(params), result_mode, stat_group_id));

Use anonymous functions instead, that's the preferred way.
I'd either write the code as local anonymous function, or simply wrap it in a local anonymous function (lambda). That's analogous to what Callback does:

```
const int a = 10, b = 10;

Thread::Run([=]{ Cout() << a *b; }); // <- Use capture by value; (If you are going to capture by reference insted, be veyy careful, as this is a thread.
```

Or

```
Thread::Run([=]{  execute_bot_in_background(bot1, a, pick(params), result_mode, stat_group_id); });
```

As a side note: If you are working with threads, and you don't need dedicated threads but worker threads, I suggest using Upp::CoWork, or Upp::AsyncWork
They are much easier to use and control, and CAN give a greater performance, as they are optimized to be used as worker threads.
Just check their docs and reference examples.

Best regards,
Oblivion

Subject: Re: I request the implementation of callback5
Posted by aftershock on Tue, 15 May 2018 18:26:14 GMT
View Forum Message <> Reply to Message

it does not seem to be work..

Instead , I had to do this

```
      Atomic wait_until;
wait_until.store ( 1, std::memory_order_relaxed );
threads[free_index].Run ( [&,a,bot1,result_mode, stat_group_id]
{
 VectorMap<String, double> params2 = pick ( params );
 wait_until.store ( 0, std::memory_order_relaxed );

 execute_bot_in_background ( bot1, a, pick ( params2 ), result_mode, stat_group_id );


} );

while ( 1 )
{
 // memory barrier for visibility semantics


 // spin wait
 if ( !wait_until.load ( std::memory_order_acquire ) )
 {
  break;
 }
}
```

I wonder if that was safe
threads[free_index].Run (THISBACK5 (execute_bot_in_background, bot1, a, pick(params), result_mode, stat_group_id));

How was params copied? Could it happen by the time pick would transfer it.. params could go out of scope faster?

By the way, using lambda's value copying is that thread safe?

Subject: Re: I request the implementation of callback5
Posted by Oblivion on Wed, 16 May 2018 07:10:38 GMT
View Forum Message <> Reply to Message

Quote:
it does not seem to be work..

Possibly a capture issue. I can't really say what's wrong until you provide the error messages. :)

threads[free_index].Run ([&,a,bot1,result_mode, stat_group_id]

Not a good practice,really. Try to avoid using default capture by reference where possible. As it captures every variable in scope (This can give you headaches when your code gets complex.).

And as I said before, If you are capturing by reference, you need to take into account the lifetime of the captured object to prevent dangling references.

Capture by reference explicitly only the variables you need. And use the capture by value mechanism for others:

threads[free_index].Run ([=,&a,&bot1,&result_mode, &stat_group_id]

By the way, using lambda's value copying is that thread safe?

Capture by value, copies (or moves where possible or explicitly stated) the parameters for each instance.
In general it is no different than ordinary copy operations, so yes.

(There are some corner cases though. I suggest reading Scott Meyers' Effective Modern C++ for information on the possible but rare complications.)

And again I suggest using CoWork if possible, as you seem to be dealing with worker threads.
A dumb, and simple example:

```
int n = 0;

CoWork co;

    for(int i = 0; i < 6; i++)
      co & [=, &n]{
            // Thread Code.
            CoWork::FinLock();
            n++;
      };
    co.Finish(); // Waits until all workers have finished their jobs. (There is also a non-blocking version: CoWork::IsFinished())
```

Best regards,
Oblivion

Subject: Re: I request the implementation of callback5
Posted by aftershock on Wed, 16 May 2018 10:03:45 GMT
View Forum Message <> Reply to Message

```
 threads[free_index].Run ( [=]//,a,bot1,result_mode, stat_group_id]
 {
  VectorMap<String, double> params2 = pick ( params ); //line 1717
//  wait_until.store ( 0, std::memory_order_relaxed );

  execute_bot_in_background ( bot1, a, pick ( params2 ), result_mode, stat_group_id );


 } );
```

```
main.cpp(1717): error C2280: 'Upp::VectorMap<Upp::String,double>::VectorMap(const
Upp::VectorMap<Upp::String,double> &)': attempting to reference a delete
    d function
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(237): note: compiler has generated
'Upp::VectorMap<Upp::String,double>::VectorMap' here
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(237): note:
'Upp::VectorMap<Upp::String,double>::VectorMap(const Upp::VectorMap<Upp::String,double>
&)': function was implicitly delet
    ed because a base class invokes a deleted or inaccessible function
'Upp::AMap<K,T,Upp::Vector<T>>::AMap(const Upp::AMap<K,T,Upp::Vector<T>> &)'
        with
        [
            K=Upp::String,
            T=double
        ]
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(215): note:
'Upp::AMap<K,T,Upp::Vector<T>>::AMap(const Upp::AMap<K,T,Upp::Vector<T>> &)': function
was implicitly deleted because a da
    ta member invokes a deleted or inaccessible function 'Upp::Index<Upp::String>::Index(const
Upp::Index<Upp::String> &)'
        with
        [
            K=Upp::String,
            T=double
        ]
```

Subject: Re: I request the implementation of callback5
Posted by Oblivion on Wed, 16 May 2018 11:25:20 GMT
View Forum Message <> Reply to Message

```
VectorMap<String, double> params2 = pick ( params ); //line 1717
```

Because params is already implicitly moved while captured.
You need to either capture it by reference, or explicitly move it (the latter is preferred).:

```
void Foo(const VectorMap<String, double>& vm)
{
 DUMP(vm);
}

CONSOLE_APP_MAIN
{
 VectorMap<String, double> params;
 params.Add("Hello world.") = 999;

 Thread().Run([=, params = pick(params)]{ // Moves params. C++14 feature (AFAIK default in
U++)
  Foo(params);
 });
}
```

Best regards,
OBlivion

Subject: Re: I request the implementation of callback5
Posted by aftershock on Wed, 16 May 2018 21:26:03 GMT
View Forum Message <> Reply to Message

No, moving does not work...

```
D:\m\upp\tradetester\main.cpp(1717): error C2280:
'Upp::VectorMap<Upp::String,double>::VectorMap(const Upp::VectorMap<Upp::String,double>
&)': attempting to reference a deleted fu
    nction
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(237): note: compiler has generated
'Upp::VectorMap<Upp::String,double>::VectorMap' here
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(237): note:
'Upp::VectorMap<Upp::String,double>::VectorMap(const Upp::VectorMap<Upp::String,double>
&)': function was implicitly deleted b
    ecause a base class invokes a deleted or inaccessible function
'Upp::AMap<K,T,Upp::Vector<T>>::AMap(const Upp::AMap<K,T,Upp::Vector<T>> &)'
        with
        [
            K=Upp::String,
```

```
        T=double
    ]
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(215): note:
'Upp::AMap<K,T,Upp::Vector<T>>::AMap(const Upp::AMap<K,T,Upp::Vector<T>> &)': function
was implicitly deleted because a data m
  ember invokes a deleted or inaccessible function 'Upp::Index<Upp::String>::Index(const
Upp::Index<Upp::String> &)'
    with
    [
        K=Upp::String,
        T=double
    ]
d:\upp-mingw-11873\upp\uppsrc\core\Index.h(222): note: 'Upp::Index<Upp::String>::Index(const
Upp::Index<Upp::String> &)': function was implicitly deleted because 'Upp::Index<Upp::
  String>' has a user-defined move constructor
```

Subject: Re: I request the implementation of callback5
Posted by mirek on Fri, 18 May 2018 12:50:15 GMT
View Forum Message <> Reply to Message

aftershock wrote on Wed, 16 May 2018 23:26No, moving does not work...

```
D:\m\upp\tradetester\main.cpp(1717): error C2280:
'Upp::VectorMap<Upp::String,double>::VectorMap(const Upp::VectorMap<Upp::String,double>
&)': attempting to reference a deleted fu
  nction
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(237): note: compiler has generated
'Upp::VectorMap<Upp::String,double>::VectorMap' here
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(237): note:
'Upp::VectorMap<Upp::String,double>::VectorMap(const Upp::VectorMap<Upp::String,double>
&)': function was implicitly deleted b
  ecause a base class invokes a deleted or inaccessible function
'Upp::AMap<K,T,Upp::Vector<T>>::AMap(const Upp::AMap<K,T,Upp::Vector<T>> &)'
    with
    [
        K=Upp::String,
        T=double
    ]
d:\upp-mingw-11873\upp\uppsrc\core\Map.h(215): note:
'Upp::AMap<K,T,Upp::Vector<T>>::AMap(const Upp::AMap<K,T,Upp::Vector<T>> &)': function
was implicitly deleted because a data m
  ember invokes a deleted or inaccessible function 'Upp::Index<Upp::String>::Index(const
Upp::Index<Upp::String> &)'
    with
    [
        K=Upp::String,
```

T=double
]
d:\upp-mingw-11873\upp\uppsrc\core\Index.h(222): note: 'Upp::Index<Upp::String>::Index(const Upp::Index<Upp::String> &)': function was implicitly deleted because 'Upp::Index<Upp::String>' has a user-defined move constructor

Hard to say without seeing the code....

The example Oblivion has posted is correct and addresses exactly this issue.

Mirek
