

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-implementantion 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

---