Subject: lambda troubles...
Posted by mirek on Sat, 03 Sep 2016 09:14:53 GMT
View Forum Message <> Reply to Message

Hi,

I am still having some issues with lambda integration and I am interested in community opinion about this...

The whole trouble is caused by two factors:

a) C++11 lambdas have 'unknown type'. That is why they have to be captured by 'catch all' constructor in std::function/Upp::Function

b) U++ originally used operator<<= overload for both Value assignment to widgets and Callback assignments.

So I have quickly learned hard way that it is not possible to use lambdas 'directly' like that lambda is directly assigned to Callback, e.g.:

```
Button b;
b <<= [=] { Foo(); };
```

because if things are overloaded for this, then catch-all constructor for lambda activates also for

```
EditString e;
e <<= "foo";
```

So, I had to make Callback and Function separate and instead introduced operator<< which 'adds lambda to callback'. Adding is in most cases better anyway.

```
Button b;
b << [=] { .... };
```

This really is quite nice, until I have found that it makes it a little bit harder to assign lambda to multiple targets. Originally this worked:

```
EditString a, b, c;
a <<= b <<= c <<= Sync();
```

but this is nonsense:

```
EditString a, b, c;
a << b << c << [=] { Sync(); };
```

(not that I have not done this mistake...)

while it is possible to write it this way:

```
EditString a, b, c;
a <<= b <<= c << [=] { Sync(); };
```

I do not like that very much. I am now thinking if this could be improved somehow... (frankly, it is possibly the last thing I would like to resolve in the new Core). One possible solution is to establish another operator for assigning lambda:

```
EditString a, b, c;
a ^= b ^= c ^= [=] { Sync(); };
```

which would either was plain assignment or 'add lambda and return it'.

Maybe it would be also possible to use '>>'

```
[=] { Sync(); } >> a >> b >> c;
```

but I guess it looks to weird...

Any thoughts?
