Subject: Re: 2022(?).2 beta
Posted by Lance on Sun, 18 Dec 2022 13:48:42 GMT
View Forum Message <> Reply to Message

Hi Mirek:

Thank you for your attention to this matter.

The following macro will perfectly pacify both GCC and CLANG.

```
#if __cplusplus > 201703L
#    define CAPTURETHISBYVALUE ,this
#else
#    define CAPTURETHISBYVALUE
#endif
```

And when using it

```
void ColorWindow::Paint(Draw& draw)
{
    auto f = [= CAPTURETHISBYVALUE]{ auto v = GetData(); };
    draw.DrawRect(GetSize(), White());
    auto v = f();
        ...
}
```

It should be easy to add support for MSVC similarly too.

This way, we only care that the U++ library can compiles with std=c++14, std=c++17, std=c++20 or later std. Whether a u++ library user decide to follow the practice so that his/her code is also multiple c++ standards compatible, or simply choose one of the standard to embrace, is not a concern of u++ library developers (like you and Klugier).

The philosophy here is: u++ can choose a stable and well supported c++ standard to embrace, but it should not limit or discourage its users from trying later standard. IMHO, comparing to package-wise c++ standard selection options(it will certainly confuse assist++ if at all doable), this kind of fix in the U++ library level is less painful.

BR,
Lance

PS:

Or probably even easier.

```
#if __cplusplus > 201703L
#    define CAPBYVALUETHIS =,this
#else
#    define CAPBYVALUETHIS =
#endif
```

Then do a find in files and replace from uppsrc root, it's almost done. I figure there are <=2 occassioins where [=] are not within a member function thus [=,this] is invalid which need be fixed after the find and replace.
