mirek wrote on Tue, 23 April 2019 10:04
That is exactly the reason why we support setting skin on startup.

See, I guess there is a certain mismatch of expectations. For me and for U++ users I care about, skinning is not that important. I just want the app look as much integrated into host as possible, do not care much above that. And most users do not switch themes 3 times per hour.

Yeah, but that only works if you expect nobody to fine tune the skin. If you are fine with the high contrast eye-strain default skin.

You could fine tune it before TopWindow starts, but TheIDE for example, sets the skin after loading the settings, so all changes done before the main window is created are lost. So maybe if proper live skinning won't be supported, TheIDE should be fixed so you can:

```
#ifdef flagMAIN
GUI_APP_MAIN
#else
void AppMain___()
#endif
{
// Ctrl::ShowRepaint(50);

 Logi() << UPP_FUNCTION_NAME << "(): " << SplashCtrl::GenerateVersionInfo(' ');

 Ctrl::SetUHDEnabled();
 Ctrl::SetDarkThemeEnabled();
 ...snip...
 LoadSkinSetting();

 // HERE IS A CLEAR POINT WHERE I CAN CHANGE THE SKIN ONCE

 // first line in ide/main.cpp where the GUI is used
```

This won't work when the user changes skins from settings, but as you said, people don't change skins that often.

But if you:

mirek wrote on Tue, 23 April 2019 10:04
Fixable, just forgot to fix it.

...we may not have a global way to change skins, but in my apps I can emulate the "OnSkinChange" skin invalidation by calling a couple of SetSkins and recreating some borders, so no big issue for me.

For SetDarkThemeEnabled, I propose something like:

void SetDarkThemeEnabled(darkSkin = Ctrl::AutodetectDarkSkin);

with

Ctrl::AutodetectDarkSkin autodetecting and behaving like old SetDarkThemeEnabled(true)
Ctrl::DarkSkin always dark, with SetDarkThemeEnabled(true) and sEmulateDarkTheme always true
Ctrl::WhiteSkin always light, with SetDarkThemeEnabled(false) and sEmulateDarkTheme always false

And bool IsSystemThemeDark() made public.