Subject: Re: U++ 2019.1.rc4 released
Posted by cbpporter on Thu, 18 Apr 2019 12:29:43 GMT
View Forum Message <> Reply to Message

OK, let's go in order.

Before the dark mode support, setting dark mode in Windows didn't work. So clearly it is not a "simple" issue of applying a system theme, because most applications ignore that windows option.

So U++ creates this new dark theme.

So my questions are:
- how do you activate dark theme in any app once, and have it work forever? Even if Windows is in light mode. Same for light mode.
- how do you detect which mode it is. I want: if (isdarkmode) setdarkmode else setlightmode. That is a nice intuitive model.
- how do you do a few basic changes to it, like SColorPaper_Write(Color(51, 51, 51)) and force this change forever?
- how do you do the same for TheIDE, because things that work in normal apps need special extra work for TheIDE.

Quote:Thats centralized model which is not extensible - that is the issue I wanted to avoid.
It looks like to me like you achieved the exact opposite, with Chameleon doing its own thing and being super hard to reliably and intuitively override. Is there anybody who ever had luck with doing anything in Chameleon except for me 10 years ago?

This invisible magic macro system is IMHO in no way superior to a non-extensible simple deign and the non-extensible bit is arguable. There is nothing non-extensible about it because you don't need to extend upon it. For years now people have been able to define incredibly diverse themes with a CSS and a bunch of pictures.

Case in point:
void Ctrl::SetSkin(void (*_skin)())

It is not clear what it does and there are 100 more ways to touch skins. I remember now, but my bazaar/Theme did not call this function and in consequence you couldn't reliably change the theme after program launch. So GUI elements and colors updated, some didn't. So I decided back then to set the them as the first thing in the program.