Subject: DarkTheme function parameters changed
Posted by mirek on Tue, 10 Sep 2024 14:29:00 GMT
View Forum Message <> Reply to Message

I have tried tuning DarkTheme function (function that "inverts" colors designed for light theme into "dark" colors) a bit as I was not happy about blue colors converted to DarkTheme. This time, I have approached it really emprically and created upptst/DarkTheme application (supposed to run in light theme) to finetune 3 relevant parameters. Would you find my current set of parameters wanting, you can try yourself and suggest different set...

File Attachments
1) Clipboard01.png, downloaded 480 times

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Thu, 12 Sep 2024 10:03:31 GMT
View Forum Message <> Reply to Message

Hi,

I wish I could help here. I tried this and could not figure out any sensible settings. My red-green colorblindness seems to make it very difficult (or impossible) to figure out what might work well for all. On the other hand, I can tell what does not work well for red-green colorblind people, so I'm happy to help on that.

For all GUI designers, I recommend reading the Wikipedia article on color blindness... all the way down to Digital design at least :)

https://en.wikipedia.org/wiki/Color_blindness

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Thu, 12 Sep 2024 14:32:18 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I've been surfing the web today about dark theme and UI coloring. Didn't find any usable algorithms for 'color inversion'.

Anyway, I found some interesting reading about dark theme and color usage here:

https://m2.material.io/design/color/dark-theme.html

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Sat, 14 Sep 2024 14:31:15 GMT
View Forum Message <> Reply to Message

I have pushed another attempt at improved DarkTheme function. Please check.

Also, upptst/DarkTheme is also updated, maybe you can find better set of constants...

(Note: I am doing all this because previous algorithm blue undo arrow icon was basically translated to white... so trying something that leaves it blue).

Mirek

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Sat, 14 Sep 2024 20:32:16 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Very, very nice! After tuning for a good while, I ended up with 0.24, 0.7, 0.16 ... but then I looked at the old code and there was within a comment 0.21, 0.72, 0.07 for physiologically correct values. I guess they are even better with this new algorithm.

Now that DarkTheme() 'color inversion' works so great, maybe a slight tuning of default TheIDE text colors (along the lines laid out in material.io Dark theme documentation) is in place. I'm thinking about:
1. replacing any saturated (default) text colors with non-saturated pastel colors
2. checking the text contrast with those colors on both light and dark background

This might offer more balanced light/dark theme experience (... maybe).

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Sat, 14 Sep 2024 21:18:58 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 14 September 2024 17:31(Note: I am doing all this because previous algorithm blue undo arrow icon was basically translated to white... so trying something that leaves it blue).

Mirek
Hi,

The undo arrow icon seems to be saturated blue. So, if you do re-coloring for the icon:

, or something similar, you should get a more balanced light/dark theme behavior. After all, blue is a very small contributor to the amount of luminance and therefore it is logical that the always dark blue becomes very light blue.

Best regards,

Tom

File Attachments
1) Screenshot from 2024-09-15 00-09-05.png, downloaded 374 times

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Sun, 15 Sep 2024 08:10:49 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 14 September 2024 22:32Hi Mirek,

Very, very nice! After tuning for a good while, I ended up with 0.24, 0.7, 0.16 ... but then I looked at the old code and there was within a comment 0.21, 0.72, 0.07 for physiologically correct values. I guess they are even better with this new algorithm.

Well.. The whole purpose is that with previous algorithm, LtRed becomes pink, Blue becomes white. Which is true here as well, with "physiologically correct" values. (The reason is simple - LtBlue is still dark color and we do not have enough range to make it light).

So it is all about compromise - after all, what matters is that colored texts remain readable while retaining reasonable color information. Warning text still needs to be red (and preferably not pink).

Mirek

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Mon, 16 Sep 2024 11:01:31 GMT
View Forum Message <> Reply to Message

Hi Mirek,

These two functions can be useful tools when working with colors, so please feel free to merge them in Color.h / Color.cpp if you like:

```
// Formula from https://www.w3.org/TR/2008/REC-WCAG20-20081211/#relativeluminancedef

double RelativeLuminance(Color color) {
 auto comp = [&] (double c){
  c /= 255;
  return (c <= 0.03928) ? c / 12.92 : pow((c + 0.055) / 1.055, 2.4);
 };
 return comp(color.GetR()) * 0.2126 + comp(color.GetG()) * 0.7152 + comp(color.GetB()) * 0.0722;
}

// Formula from https://www.w3.org/TR/2008/REC-WCAG20-20081211/#contrast-ratiodef

double ContrastRatio(Color c1, Color c2) {
 double rl1 = RelativeLuminance(c1);
 double rl2 = RelativeLuminance(c2);
 return (max(rl1, rl2) + 0.05) / (min(rl1, rl2) + 0.05);
}
```

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Tue, 17 Sep 2024 07:10:55 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 16 September 2024 13:01Hi Mirek,

These two functions can be useful tools when working with colors, so please feel free to merge them in Color.h / Color.cpp if you like:

```
// Formula from https://www.w3.org/TR/2008/REC-WCAG20-20081211/#relativeluminancedef

double RelativeLuminance(Color color) {
 auto comp = [&] (double c){
  c /= 255;
  return (c <= 0.03928) ? c / 12.92 : pow((c + 0.055) / 1.055, 2.4);
 };
 return comp(color.GetR()) * 0.2126 + comp(color.GetG()) * 0.7152 + comp(color.GetB()) * 0.0722;
}
```

```
// Formula from https://www.w3.org/TR/2008/REC-WCAG20-20081211/#contrast-ratiodef

double ContrastRatio(Color c1, Color c2) {
 double rl1 = RelativeLuminance(c1);
 double rl2 = RelativeLuminance(c2);
 return (max(rl1, rl2) + 0.05) / (min(rl1, rl2) + 0.05);
}
```

Best regards,

Tom

Why not... added.

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Tue, 17 Sep 2024 15:00:28 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 14 September 2024 22:32Now that DarkTheme() 'color inversion' works so great, maybe a slight tuning of default TheIDE text colors (along the lines laid out in material.io Dark theme documentation) is in place. I'm thinking about:

Default scheme should always be "host". However, I can add alternative dark theme, if you supply one. It is quite easy, check

CtrlLib/Ch.cpp:716

Mirek

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Tue, 17 Sep 2024 18:31:45 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 17 September 2024 18:00Tom1 wrote on Sat, 14 September 2024 22:32Now that DarkTheme() 'color inversion' works so great, maybe a slight tuning of default TheIDE text colors (along the lines laid out in material.io Dark theme documentation) is in place. I'm thinking about:

Default scheme should always be "host". However, I can add alternative dark theme, if you supply one. It is quite easy, check

CtrlLib/Ch.cpp:716

Mirek

I was actually just thinking about TheIDE accent colors for code editing and other views... just to adjust the dark variants to view nicely on dark backgrounds. But yes, this involves dark theme tuning as well to get contrast optimized.

I remember working on Win32 emulated dark theme colors. The trick was (and I think still is) that many definitions come from platform and mostly only colors need to be adjusted for the dark theme. I recall that the color changes had to be written in-line within ChHostSkin(). If I just create a function called e.g. ChMyCustomSkin(), similar to e.g. ChDarkSkin(), I will not get platform shapes/styles for widgets, but something else instead. Therefore, some reorganization would be helpful:

It would be nice to have theme loading clearly split to two parts:

1. Load system colors (custom or platform) and store with "SColor*_Write();" functions like before.
2. Load shapes/styles from host like in ChHostSkin() (or load custom shapes/styles), and combine with preloaded SColor*() colors for a complete theme.

Also live theme changes would be nice... We have talked about this before and at that time you warned about a potential problem that arises with some controls caching colors internally. Still, it would be an interesting improvement. (I can even imagine a generic Theme(Color)EditorDialog after live theme changes feature has been introduced...)

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Tue, 17 Sep 2024 21:45:19 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 17 September 2024 20:31
It would be nice to have theme loading clearly split to two parts:

1. Load system colors (custom or platform) and store with "SColor*_Write();" functions like before.
2. Load shapes/styles from host like in ChHostSkin() (or load custom shapes/styles), and combine with preloaded SColor*() colors for a complete theme.

You cannot load just shapes. All 3 hosts we use paint the whole widget (think images of widgets).

Mirek

Subject: Re: DarkTheme function parameters changed

Posted by Tom1 on Wed, 18 Sep 2024 19:11:07 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 18 September 2024 00:45Tom1 wrote on Tue, 17 September 2024 20:31
It would be nice to have theme loading clearly split to two parts:

1. Load system colors (custom or platform) and store with "SColor*_Write();" functions like before.
2. Load shapes/styles from host like in ChHostSkin() (or load custom shapes/styles), and combine with preloaded SColor*() colors for a complete theme.

You cannot load just shapes. All 3 hosts we use paint the whole widget (think images of widgets).

Mirek
>
OK, I see. So, basically we're stuck with the colors given by host, right? I mean for buttons, scroll bars and more...

Is it also true that if full control of colors is desired, the widgets need to be created internally like ChClassicSkin() and ChStdSkin() do?

BR, Tom

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Wed, 18 Sep 2024 19:59:25 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 18 September 2024 21:11mirek wrote on Wed, 18 September 2024 00:45Tom1 wrote on Tue, 17 September 2024 20:31
It would be nice to have theme loading clearly split to two parts:

1. Load system colors (custom or platform) and store with "SColor*_Write();" functions like before.
2. Load shapes/styles from host like in ChHostSkin() (or load custom shapes/styles), and combine with preloaded SColor*() colors for a complete theme.

You cannot load just shapes. All 3 hosts we use paint the whole widget (think images of widgets).

Mirek
>
OK, I see. So, basically we're stuck with the colors given by host, right? I mean for buttons, scroll bars and more...

No, we can choose any colors we wish - but we need to draw our widgets too... but if you have

checked to code reference I sent, it is doing exactly that there...

DarkTheme in windows is currently sort of hack - we apply DarkTheme on light theme widgets (because that is what API we currently use is returning).

Mirek

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Wed, 18 Sep 2024 20:37:30 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 18 September 2024 22:59
DarkTheme in windows is currently sort of hack - we apply DarkTheme on light theme widgets (because that is what API we currently use is returning).

Mirek

Yes, I know it all too well from 2022:

https://www.ultimatepp.org/forums/index.php?t=msg&th=106 57&goto=57856&#msg_57856

Well, I'll think about this theming, and let you know if I come up with anything potentially useful.

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Sun, 06 Oct 2024 12:46:52 GMT
View Forum Message <> Reply to Message

Thanks to work of both of Mirek and Tom1, Dark Theme now works very naturally on gnome.

Here is a small feature request that may improve user experience on gnome or possible x11 as a whole. But if it's too complicated or unworthy, just ignore it.

On more recent gnome (or at least ubuntu version of gnome), we can quickly switch between darkstyle and normal style using the dropdown menu:

Native applications respond to the changes immediately while, for theide, we have to restart it after theme changes. This makes theide and upp appliations stand out in a negative way.

I was told dconf command tool can monitor the changes. Indeed

$dconf watch /org/gnome/desktop/interface/color-scheme

will report the changes. Or underneath, gio, can be used somehow to access the settings or possibly even subscribe to certain changes, which I don't know how.

If upp can listen to the event and request all Ctrl to Refresh (hierachically) themselves, and all SColorXXX returns a reference (to a static variable who has application-long life time which will by updated by Upp at theme-changes), a upp application can behave more similar to a native gnome/x11 application.

However, all user programs alos need to change accordingly to use Color& instead of Color to store SColorXXXs that they use in their own chameleon style. But this won't break their programs catastrophically even if they don't: any part that they call SColorXXX on the site will be reflectly after first Refresh, while the part they stored a copy of SColorXXX, in, for example, a Chameleon style, will only be corrected after a restart.

More can be involved than just color, eg, icon,etc. But it could be a simple , albeit tedious, work for Mirek.

File Attachments

1) Screenshot from 2024-10-06 08-17-42.png, downloaded 279 times

---

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Sun, 06 Oct 2024 19:35:44 GMT
View Forum Message <> Reply to Message

The problem with switching the theme is actually a problem of all those stored colors in all widgets and data. E.g. what to do with AttrText that has some dark adjusted ink stored in it.

U++ can actually switch the theme with immediate response (just check upptst::ChStyle, it does just that - but notice that background in example ArrayCtrl does not change), the reson for restart is actually to make developer's live easier.

But OK, I can add "SkinChanged" virtual method and leave the immense work of reskining color in developer maybe...

Mirek

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Sun, 06 Oct 2024 19:37:56 GMT
View Forum Message <> Reply to Message

Lance wrote on Sun, 06 October 2024 14:46
If upp can listen to the event and request all Ctrl to Refresh (hierachically) themselves, and all SColorXXX returns a reference (to a static variable who has application-long life time which will by updated by Upp at theme-changes), a upp application can behave more similar to a native gnome/x11 application.

However, all user programs alos need to change accordingly to use Color& instead of Color to store SColorXXXs that they use in their own chameleon style. But this won't break their programs catastrophically even if they don't: any part that they call SColorXXX on the site will be reflectly after first Refresh, while the part they stored a copy of SColorXXX, in, for example, a Chameleon style, will only be corrected after a restart.

That is not enough, look at this:

CtrlLib/ArrayCtrl.h:686

```
 ArrayCtrl& EvenRowColor(Color paper = Blend(SColorMark, SColorPaper, 220), Color ink =
SColorText);
```

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Sun, 06 Oct 2024 20:32:57 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 06 October 2024 15:35The problem with switching the theme is actually a problem of all those stored colors in all widgets and data. E.g. what to do with AttrText that has some dark adjusted ink stored in it.

U++ can actually switch the theme with immediate response (just check upptst::ChStyle, it does just that - but notice that background in example ArrayCtrl does not change), the reson for restart is actually to make developer's live easier.

But OK, I can add "SkinChanged" virtual method and leave the immense work of reskining color in developer maybe...

Mirek

Thanks for the information. Please don't add "SkinChanged". I was thinking maybe add a layer of indirection (Store reference to a static Color variable instead of a copy of it in Chameleon Style). If it is more complicated than that, it's not worth it.

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Sun, 06 Oct 2024 20:34:44 GMT

mirek wrote on Sun, 06 October 2024 15:37
That is not enough, look at this:

CtrlLib/ArrayCtrl.h:686

ArrayCtrl& EvenRowColor(Color paper = Blend(SColorMark, SColorPaper, 220), Color ink = SColorText);

Thanks!

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Sun, 06 Oct 2024 20:52:09 GMT

mirek wrote on Sun, 06 October 2024 15:37That is not enough, look at this:

CtrlLib/ArrayCtrl.h:686

ArrayCtrl& EvenRowColor(Color paper = Blend(SColorMark, SColorPaper, 220), Color ink = SColorText);

Please educate me if I am wrong.

What are SColorMark, SColorPaper, SColorText? Are they some Color variable in ArrayCtrl's Chameleon Style?

When a user code called

```
EvenRowColor();
```

Something like this happen (conceptually)

```
push ink = SColorText;
push paper = Blend(SColorMark, SColorPaper, 220)
push this
call ArrayCtrl::EvenRowColor
```

If SColorText, SColorMark, SColorPaper are all refreshed after the theme change, EvenColor should receive correct color parameters.

So if instead of store a copy of SColorxxx(), let it store a reference/pointer of SColorxxx() instead. And all SColorXXX() should return Color& instead of Color.

Will this be adequate?

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Sun, 06 Oct 2024 22:26:01 GMT
View Forum Message <> Reply to Message

Lance wrote on Sun, 06 October 2024 22:52mirek wrote on Sun, 06 October 2024 15:37That is not enough, look at this:

CtrlLib/ArrayCtrl.h:686

ArrayCtrl& EvenRowColor(Color paper = Blend(SColorMark, SColorPaper, 220), Color ink = SColorText);

Please educate me if I am wrong.

What are SColorMark, SColorPaper, SColorText? Are they some Color variable in ArrayCtrl's Chameleon Style?

When a user code called

```
EvenRowColor();
```

Something like this happen (conceptually)

```
push ink = SColorText;
push paper = Blend(SColorMark, SColorPaper, 220)
push this
call ArrayCtrl::EvenRowColor
```

If SColorText, SColorMark, SColorPaper are all refreshed after the theme change, EvenColor should receive correct color parameters.

So if instead of store a copy of SColorxxx(), let it store a reference/pointer of SColorxxx() instead. And all SColorXXX() should return Color& instead of Color.

Will this be adequate?

It could easily be a reference, but it does not solve the problem, unless you add somewhat complicate mechanism where the "Blend" part is somehow encoded. I mean, you can make SColorMark work, SColorPaper work, but what is actually stored is some other color, a mix of those two.

It is too clumsy and 'undefined' IMO for the rapid development.

Mirek

Subject: Re: DarkTheme function parameters changed

Posted by Lance on Sun, 06 Oct 2024 23:59:18 GMT
View Forum Message <> Reply to Message

Do you mean that we can have SColorPaper,SColorMark theme-sensitive, but there is no guarantee Blend(SColorPaper, SColorMark, 220) will make any sense in all possible themes we may throw at it?

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Mon, 07 Oct 2024 06:51:10 GMT
View Forum Message <> Reply to Message

Hi,

Sorry to interfere here, but I would not mind having both theme dependent SColors and "SkinChanged" virtual method called on Ctrls when system theme changes. When the theme changes, everything obviously needs to be Refresh()ed, which could be the default behavior of "SkinChanged". But that is not sufficient, as in some cases content is cached in pre-rendered images which are not aware of the changed theme. Therefore, "SkinChanged" is needed to re-render everything followed by Refresh().

Is there a way to run a test on this? Maybe a branch for testing the idea?

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Mon, 07 Oct 2024 07:40:28 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 07 October 2024 08:51Hi,

Sorry to interfere here, but I would not mind having both theme dependent SColors and "SkinChanged" virtual method called on Ctrls when system theme changes. When the theme changes, everything obviously needs to be Refresh()ed, which could be the default behavior of "SkinChanged". But that is not sufficient, as in some cases content is cached in pre-rendered images which are not aware of the changed theme. Therefore, "SkinChanged" is needed to re-render everything followed by Refresh().

Is there a way to run a test on this? Maybe a branch for testing the idea?

Best regards,

Tom

Not yet. After 2024 release, I can add the changes to the master - however for forseeable future, we will need "allow theme changes" flag/function call.

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Mon, 07 Oct 2024 07:43:56 GMT
View Forum Message <> Reply to Message

Lance wrote on Mon, 07 October 2024 01:59Do you mean that we can have SColorPaper,SColorMark theme-sensitive, but there is no guarantee Blend(SColorPaper, SColorMark, 220) will make any sense in all possible themes we may throw at it?

I mean that Blend(SColorPaper, SColorMark, 220) has to be evaluated at some point. From developer perspective, the most logical point is when he calls Blend. It this was supposed to work without SkinChanged, we would need to do something like changing all colors stored in widgets to functions. I do not think that is a good idea...

Mirek

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Mon, 07 Oct 2024 08:36:10 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 07 October 2024 10:40Not yet. After 2024 release, I can add the changes to the master - however for forseeable future, we will need "allow theme changes" flag/function call.
Thanks! Looking forward to both of these events!

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Mon, 07 Oct 2024 11:57:22 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 07 October 2024 03:43

I mean that Blend(SColorPaper, SColorMark, 220) has to be evaluated at some point. From developer perspective, the most logical point is when he calls Blend. It this was supposed to work without SkinChanged, we would need to do something like changing all colors stored in widgets to functions. I do not think that is a good idea...

Mirek

```
//header
struct SysColors
{
   Color paper;
   Color ink;
   Color mark;
   Color highlight;
   ...
   void RefreshColors();

   ...

   static SysColors scs;

private:
   SysColors(){ RefreshColors(); }
};
Color& SColorPaper = SysColors::scs.paper;
Color& SColorMark  = SysColors::scs.mark;
//cpp
SysColors SysColors::scs;
```

Then

Blend(SColorPaper, SColorMark, 220)

will be evaluated everytime with refreshed System Colors.

It will involve a lot of tedious work of changing copied color values to references to variables of program life time. Other than that, it should not present unnecessary work on upp users.

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Mon, 07 Oct 2024 12:03:23 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 07 October 2024 04:36mirek wrote on Mon, 07 October 2024 10:40Not yet. After 2024 release, I can add the changes to the master - however for forseeable future, we will need "allow theme changes" flag/function call.
Thanks! Looking forward to both of these events!

Best regards,

Tom

I am mostly on DarkStyle, means I don't need to switch back and forth. It's a matter of u++ user experience.

Thanks for your great work BTW.

Subject: Re: DarkTheme function parameters changed
Posted by Tom1 on Mon, 07 Oct 2024 12:22:15 GMT
View Forum Message <> Reply to Message

Lance wrote on Mon, 07 October 2024 15:03I am mostly on DarkStyle, means I don't need to switch back and forth. It's a matter of u++ user experience.
Well, so am I. :)

Yes, it's all about user experience: Many of my clients sometimes run 24/7 on vehicles and there it is very useful to be able to switch between dark and light modes (I mean the entire Windows theme) when the Sun rises and sets without need to shut down and restart our software.

Best regards,

Tom

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Mon, 07 Oct 2024 12:48:33 GMT
View Forum Message <> Reply to Message

Lance wrote on Mon, 07 October 2024 13:57mirek wrote on Mon, 07 October 2024 03:43

I mean that Blend(SColorPaper, SColorMark, 220) has to be evaluated at some point. From developer perspective, the most logical point is when he calls Blend. It this was supposed to work without SkinChanged, we would need to do something like changing all colors stored in widgets to functions. I do not think that is a good idea...

Mirek

```
//header
struct SysColors
{
  Color paper;
  Color ink;
  Color mark;
  Color highlight;
  ...
  void RefreshColors();
```

```
  ...

  static SysColors scs;

private:
  SysColors(){ RefreshColors(); }
};
Color& SColorPaper = SysColors::scs.paper;
Color& SColorMark  = SysColors::scs.mark;
//cpp
SysColors SysColors::scs;
```

Then

Blend(SColorPaper, SColorMark, 220)

will be evaluated everytime with refreshed System Colors.

No.

```
ArrayCtrl ctrl;
ctrl.EvenRowColor();
```

it gets evaulate once, at this point. If SystemColors change, it is still evaluated for the old values. Changing values to references does not really help, you would at least need to call EvenRowColor again.

And pls note that this is just one example. What about e.g.

```
ctrl.Add(AttrText("Hello there!").Ink(IsDarkTheme() ? Blend(White(), LtBlue()) : Blue());
```

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Mon, 07 Oct 2024 14:49:35 GMT
View Forum Message <> Reply to Message

I see the challenge now.

We need something like theme-dependent color.

With regard to Blend(), would it ever need more than 2 colors?

I have a hackish solution.

```
class DynColor{
  struct _combo{
     Color c;
     char  f[4];
  };

  operator Color()const{
    switch(f[3] & 3){
    case 0: // the 64 bits are a Color*
       return * reinterpret_cast<const Color*>(this);
    case 1: // a normal color hold at data.c
       return c;
    case 2: // Blend, with f[0], f[1] the index
          // of System colors, hopefully
          // we don't have more than 256 of them
          // , and f[2] hold the third paramenter of Blend
      return Blend(GetSysColorFromIndex(f[0],
                GetSysColorFromIndex(f[1]),
                f[2]);
     case 3: // let's handle the
//ctrl.Add(AttrText("Hello there!").Ink(IsDarkTheme() ? Blend(White(), LtBlue()) : Blue());
             // case here
         // theme dependant blend
         return IsDarkTheme() ?
              Blend( treat_c as byte[4], and store blend information there) :
              Blend(GetSysColorFromIndex(f[0],
                GetSysColorFromIndex(f[1]),
                f[2]);

    }
  }

private:
  combo data;
};
```

Just a prototype for conception. We need to consider endianness at least.

Now, we can

ctrl.Add(AttrText("Hello there!").Ink(ConstructDynColor(DarkBlender{...}, NormalBlender{...}));

Yes I am aware this requires changes to AttrText code. I can imagine it's quite involving.

Also this will require everywhere that requires dynamic color to use 64-bit color, a double in storage space.

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Mon, 07 Oct 2024 15:24:28 GMT
View Forum Message <> Reply to Message

ctrl.Add(AttrText("Hello there!").Ink(IsDarkTheme() ? Blend(White(), LtBlue()) : Blue());

theme-dependant color should really start from theme defined base colors. If user code ever calls IsDarkTheme(), it's really his/her responsibility to override ThemeChanged virtual function to handle these.

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Mon, 07 Oct 2024 15:34:23 GMT
View Forum Message <> Reply to Message

On a second thought, ThemeChanged virtual function might be the easier and less expensive way to go.

Subject: Re: DarkTheme function parameters changed
Posted by Lance on Wed, 09 Oct 2024 21:39:20 GMT
View Forum Message <> Reply to Message

Maybe we can do dynamic color within the current Color. I don't know the full picture, but here are some thoughts:

1. upto 254 System colors are store in a c array that is indexable with a integer i (0<=i<=254); These are, of course, SColorPaper, SColorInk, SBlack, SWhite, etc.

2. End user(programmer) are deprived of alpha value of 255. With alpha=255
  a. if r=g=b=255, that's the Null Color;
  b. If g=b=255, r<255, r is the color index of a system color;
  c. if b<=254, the Final Color = Blend(ColorFromIndex(r), ColorFromIndex(g), b);

3. If a programmer's Color choice conflicts with dynamic color ones, a warning will be logged but a color close enough will be silently supplied instead.

4. Color are finalized on the site. So AttrTxt, etc, only calculate the final Color before it actually renders text.

5. UPP supports light theme(normal theme), dark theme out of box, free of charge. If a programmer wants his/her program sticks to light theme, sticks to dark theme, stick to host theme (between dark and non-dark, any non-dark host theme will be interpreted to light theme by upp, before further themes are developed, if ever). Remove the chamalion Style * style;(a further 8 byte reduction on x64 platforms) from Ctrl definition, but add a GetChStyle()const virtual function, which will return light theme, dark theme, or host-dependant theme according to programmers' choice. If anyone want to Style a Ctrl, he/she should override GetChStyel() to return pointers to

his/her own full set of theme values. Don't use, don't pay.

5. Icon will remain to be an issue.

Subject: Re: DarkTheme function parameters changed
Posted by mirek on Wed, 13 Nov 2024 22:54:35 GMT
View Forum Message <> Reply to Message

https://www.ultimatepp.org/forums/index.php?t=msg&goto=6 1087&#msg_61087
