Subject: Re: Any function to draw gradient color?
Posted by piratalp on Mon, 24 Sep 2007 15:22:49 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 23 September 2007 05:47ff / ss

Anyway, I have had a look at it.

First, is there a way how to obtain these values from OS? Would be nice set of complementary Color values, IF it is possible to have a reasonable emulation when not available (e.g. in Linux). If not, this should be hidden in ribbon emulation:

Second you do not really need this palette in U++ to get things work. You can store them into Value for specific chameleonized elements IMO.

Mirek

Hehe, so we're the only ones having a fff color table? :s

Well, going to the questions.. There is for sure a way to retrieve them, how? I DON'T KNOW, I never liked windoze internals.. but they are set by XP when you change color scheme (usually Blue -default-, silver and olive) and ProfessionalColorTable always reflect correct colors acording to system palette..
Of course there is _very_ reasonable emulation and it is already done and working (and looking nice by using current limited set of SColor functions. I called it ChSystemPalette(). OTOH final Office 2007 Blue palette is harcoded in program, so it works in Linux, Mac, Windows, BSD.. no matter the platform, Office 2007 comes with its own set of palettes independent from system colors.. I plan to make other Office 2007 palettes and custom ones with very nice look but that will be when I have a bit of time, now I'm very busy chameleonizing widgets that were harcoded until now..

For the second.. Yes, it was exactly what I was doing but it had some cons, major one the need to have 8 set of values for each existing widget, and the need to add a new set each time a new widget needs to be chameleonized with this kind of skin, so I decided to drop all thos Values from ChPalette and replace them with a set of colors for: Face (Normal, Highlight, Pressed & Disabled states) wich are used for practically any widget that does not have _very_ customized look, the rest have their own set of palette colors (e.g. Buttons -including Ok & Cancel ones-, Menus, ScrollBars and ProgressBars)

It sounds a bit more complicated than it is, so let's illustrate it:

What I was doing (sth like what you proposed):

Quote:
```
struct ChPalette {
  Value Background; // Used for window bg
  Value Text; // Used as global text color
```

```
   Value Button; // Normal button state
   Value ButtonText;
   Value ButtonHighlight;
   Value ButtonHighlightText;
   Value ButtonPressed;
   Value ButtonPressedText
   Value ButtonDisabled;
   Value ButtonDisabledText

   // These last 8 repeated for every existing widget
}
```

This way I had to hardcode widget painting INSIDE palette and add lots of members to the struct, so I decided to refactor it and look more like ProfessionalColorTable (fff)

Quote:

```
struct ChPalette {
   Color FaceLight;
   Color FaceMedium;
   Color FaceNormal;
   Color FaceDark;
   Color FaceBorder;
   Color FaceText;

   // So with this kind of palette it is the skin wich decides how widgets are rendered
   // This set of colors repeat changing Face with Highlight, Pressed & Disabled and are used for every widget, except the ones I mentioned above, for which this set of 4-state * 6 components is repeated
}
```

Hope it is more clear now...

Regards