Subject: Re: Global style changes using Chameleon...
Posted by mrjt on Fri, 11 Apr 2008 16:09:30 GMT
View Forum Message <> Reply to Message

I'll add what I know to this, since your guide idea is a good one.

I have a feeling that the XXX_Write functions are not meant to be comprehensive, just present where necessary for matching native themes.

The .Write() function should be present on every Chameleon style (they are all inherited from ChStyle), so I'm not sure why you need the const_cast<> stuff. Is there an example where this doesn't work?. All the Write() function does is return a non-const reference to the style so that you can edit it.

A stightly more concise version of your button example for instance:

```
Button::Style &button_style = Button::StyleNormal().Write();
button_style.look[0] = MyButton::Get(0);
button_style.look[1] = MyButton::Get(1);
button_style.look[2] = MyButton::Get(2);
button_style.look[3] = MyButton::Get(3);
button_style.textcolor[0] = White;
button_style.textcolor[1] = White;
button_style.textcolor[2] = White;
button_style.textcolor[3] = Gray;
```

Quote:I think your XXX::Style object needs to remain instantiated for the duration of your application... I typically handle this by creating these objects in main, allowing them to go out of scope when main() ends.  Perhaps someone may correct me if I'm wrong.
You are correct. Controls store a pointer to the object, so it should not run out of scope. My preferred solution is this (for single ctrls):

```
const EditField::Style &MyEditStyle() {
    static EditField::Style mystyle = EditField::StyleDefault();
    mystyle.focus = SYellow();
    return mystyle;
}

// Somewhere else
username.SetStyle(MyEditStyle());
```

This is more-or-less what the CH_STYLE macro does I believe.

James