Subject: Re: Styles and Widgets
Posted by mirek on Thu, 15 Mar 2007 11:11:11 GMT
View Forum Message <> Reply to Message

WebChaot wrote on Thu, 15 March 2007 04:14Hi Mirek!

>>> It is not really too much different.

OK. Then I did not realize how styles work. I'm not a very experienced upp-programmer (but have programming background in other languages) - so I have to learn from sources and examples. That worked quite good until now. But I do not understand the styles example

In past I created some classes based on upp standard widgets and do my own paint-methods. Thats boring, because after each update I have to update all these classes.

With styles - I guess - I can use standard widgets and only assign new paint-methods to that widgets on application startup (without writing own classes). Thats the theory. In practice I do not understand, how ChEllipse, EllipseLook(), MyLookFn() and INITBLOCK work together. Maybe there is another piece of code somewhere in this forum or in any other project?

What I need to do is to make an office 2003 or 2007 style to all widgets, if possible. Will I have to create each widget new and place my code in each paint-method or would there be a better way?

Would be happy about every information about this topic,

WebChaot.

PS: I'm not sure, if I can solve this problem. But if I finished my office-like widgets (NavigationBar, movable Toolbars, ...), I would share them here in forum, if someone would need them too.

Ah, now I see the problem. I thought you have been using first chameleon iteration (but you did it by overriding Paint).

Well, the important thing to understand about chameleon is that it is using "Value" to represent visual appearance.

Each widget is usually represented by one or more rectangular areas. Each such area has associated Value, or array of Values to define its appearance (array because area can have more states).

E.g. button has single rectangle that covers it all. It has several Values for normal, pushed etc states. Button::Paint chooses Value based on current state and calls ChPaint to paint the area of button (note that button text (and/or image etc..) is not a part of this process, it is applied later).

Now what gets painted depend on the Value. Value can contain anything and Chameleon allows the client code to register "value painter" based on its Value. Chameleon also supports two basic types - Image and Color - than can be used to paint area.

Now Image - there is important to know, that how Image is applied to area is ruled by positions of two reference points (HotSpot and 2ndSpot) that can be designed in Icon designer. Basically, you use them do designate which areas have to be stretched.

This way appearance of area is parametrized. Of course, more parameters than that are needed to customize widgets; all are stored in those Styles.

Mirek
