Subject: Re: Chameleon again
Posted by cbpporter on Mon, 18 Jan 2010 08:34:46 GMT
View Forum Message <> Reply to Message

Well for starters let us ignore system skins and just consider custom skins. For every widget you theoretically have a structure that describes how it should be painted. I'll use Button as an example. I has 4 states: normal, focused, pushed and disabled. For every state you would like to have a way to draw it without having Button be aware of the intimate details of rendering. So you use images. I'll use this very ugly image:

The two red spots are the hot spots. This divides the image into 9 areas. This image will be applied to all sizes of Button. The hot spots describe which areas will be resized. In this image, the cyan areas will remain constant size, the orange areas will be resized in one dimension, and the gray/purple one will be resized in both dimensions. Resizing algorithm is smooth and uses a filter which is appropriate for making good looking pictures while resizing. You will never get a blocky image, but sometimes you get a blurry one.

Assign this image to look[0] for Button and you will get the result described above. But you can assign other things. Color is an obvious example, but for Windows another kind of data is used. This data is obtained from the system and allows U++ applications to look almost 100% native (unfortunately no animations for Vista and Win 7). This System skins are not images and do not have hot spots. It is more like "take this are and please fill it with the look of widget X in state Y". It is a lot more complicated than this. And there was probably a healthy amount of guess work and trial and error involved until look under Windows got as good as it is now.

Under Linux there is a different mechanism which I believe instantiates the equivalent Gtk widget and obtains the correct look.

Now the internals of Chameleon are ugly and nobody except the one who wrote them understands them 100%. That part could be indeed improved, but it would be a pretty big task. The structures that provide the looks for all the widgets are easy to work with once you understand them, but there are a lot of hidden rules. If field a is present, but not b and c, then use a, otherwise use b for this and c for that. There is zero documentation for this right now (sorry). You can look over Skulpture code to see how the look is changed for most widgets.

This is just a short intro. You can post more questions and I'll try to answer the best I can.

File Attachments

1) Untitled4.png, downloaded 1381 times