Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Thu, 11 Feb 2010 00:28:12 GMT
View Forum Message <> Reply to Message

Hello,
I am back with some details about my thesis and how to integrate my efforts in U++.
I think having an unified API across different platforms, something like API provided by uxtheme from Windows would simplify Chameleon part of U++ which right now it's somehow complicated and code is messy (I'm talking here about look and feel acquisition not ChStyle).
I know that in U++ the rule is to avoid dynamic linking but I think we may do an exception if with that we acquire integration with Gnome, KDE, XFCE, Maemo using single executable and 2 *.so for gtk+ and qt. If the user is not running any of these environments the application will use U++ look. However it's not an obligation to use a so lib, if someone know that his application will run only on (i.e.) Gnome he may link statically with so (lib) responsible for application look. I've named this part (so/dll) Camouflage (you know Chameleon->Camouflage) and this part is what I'm willing to do for my graduation thesis (I think I have 3 months to finish it). If everything goes as I planned at the end we will have only one single ChHostSkin (see ChWin32, ChGtk) function, data acquisition being done by my library. About applications performance, I think that will not affect application performance because look is acquired only once at start-up (as now with ChGtk/ChWin32 implementation) and only when system theme get changed.
Following is an image with some details.

The problem I have now and I have to solve it quickly is the API.
It have some requirements.
- must be accessible through a C interface (dynamic link requirement)
- must do more than uxtheme is doing, it should provide also hotspots for images
- must avoid memory allocation/deallocation because of U++ allocator and also to avoid memory leaking (allocation on lib and freed by U++)
- must not depend on U++ (or should?), using it's types (i.e. Draw, ImageBuffer) because that would bloat it making it a huge library

Some possible API's:

```
// implement uxtheme API but without the hassle of getting a handle for control (OpenThemeData)
// and instead of sending to DrawThemeBackground a HDC to send an unsigned char*
// which could be easily converted to RGBA (or ImageBuffer, etc.)
enum CamoParts { CP_CheckBox, CP_PushButton, CP_RadioButton, ... };
enum CamoPartState { PB_Defaulted, PB_Disabled, PB_MouseHover, PB_Normal, PB_Pressed, ...};
void DrawThemeBackground(unsigned char* rgba, // image buffer, always RGBA (LittleEndian order?)
                         CamoParts cp,      // CP_PushButton
                         CamoPartState cps, // PB_Normal
                         int cx, int cy     // size of image, it's important for getting correct image buffer too
                         );
```

```
// each possible part as an enum
enum CamoParts { CP_CheckBox, CP_PushButton, CP_ScrollBarBackwardStepper,
CP_ScrollBarThumb...};
enum CamoState {
   Normal,
   Default,
   Pressed,
   Focused,
   Hot, // MouseHover
   Disabled
};
void drawPart(unsigned char* rgba, // image buffer, allocated by u++, freed by u++
            CamoParts cp,      // CP_ScrollBarThumb
            CamoState cs,      // Normal
            int cx, int cy);   // size of image
```

If someone have a smart idea I would appreciate that.

L.E.: I forget to put hotspots in API

```
struct CamoPoint
{
   dword x, y;
};
void drawPart(unsigned char* rgba, // image buffer, allocated by u++, freed by u++
            CamoParts cp,      // CP_ScrollBarThumb
            CamoState cs,      // Normal
            int cx, int cy,    // size of image
            CamoPoint* hotspot1,
            CamoPoint* hotspot2);
```

After call to drawPart the rgba will contain control image and hotspot1/2 will contain hotspots for respective control.

File Attachments

1) drawing.png, downloaded 897 times