Subject: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Wed, 11 Nov 2009 17:27:12 GMT
View Forum Message <> Reply to Message

I've talked here a bit about my degree thesis. Now it's time to start working on. I will be glad to receive any feedback about implementation, maybe if i get it done very well it may replace a part of upp chameleon which IMHO it's a bit messy, think if you have to start doing chameleon stuff for a new system, from where you start? There's no standard interface you should implement so I'm going to work on an interface which should be implemented on any system, debugging it easy and having a base from where you start when implementing it on a new system, say MacOSX.
Here I'm presenting some ideas of how it should work, pro and cons about some implementations ways. More to come as I think about it...

Name: Camouflage
Language: C/C++
Legend: + pro, - cons, ~ between pro and cons

Pro/Cons:
C Language:
+ may be used easily from other languages
+ runtime linking (dlopen,LoadLibary, etc.)
- poor standard library

C++ language:
+ OOP
~ standard library
~ no runtime linking using standard methods(possible if exporting C interfaces)

Library usage:
1. U++ linked with library at runtime and library is linked statically(because C++) with KDE(QT)
2. U++ static linked with library and library linked dynamically with GTK+, WINAPI, Cocoa*(*is that possible?)
3. U++ static linked with library which is static linked to KDE, GTK+, WINAPI, Cocoa

Explanations for library usage:
1. In case one i have to provide a library for KDE and one for GTK, the program may use gtk or kde lib based on current environment
(this is related to Linux, because other systems doesn't provide multiple Desktop Environments)
+ one program, many interfaces (gtk, kde, x11)
- multiple libraries, it may be a bit complex and the program will depend on these dll(so) (but portability is achieved)
- runtime linkage is not as fast as static linkage(compile time), but it's used on other platforms too to provide portability(i.e. Chameleon in Windows)
How it work: Program will check current environment if it's gtk then load camouflage lib bassed on gtk, if kde then load kde based lib else use only X11 for widgets draw

2. The case two is(should be) the actual U++ state, but it lack KDE style (and Cocoa, but that's another story)

IIRC U++ is static linked with GTK, you may use it without gtk(X11 only) with NOGTK flag.
- will miss KDE interface, because KDE is C++ and we cannot load C++ libs because dlopen doesn't support it and it's not possible because C++ name mangling
+ it will be possible to use gtk interface or X11 only based on program arguments, so you may have a single program for both gtk and X11(only)
How it work: if gtk is available the program with use it else it will use only X11 and will be possible to control gtk or X11 usage based on parameters sent to program
i.e. ./myprogram --X11-view # and upp lib will not load gtk libs

3. - loosing portability because it will depend directly on available libraries
- multiple binaries for program on one platform(gtk, kde, X11)

After having these pro and cons I'm thinking to make it using second approach (U++ way), but that means i will have to drop KDE support or if someone have an idea how to load a static member function of a C++ class(QApplication::style()) then the problem is like solved.

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Wed, 11 Nov 2009 18:08:58 GMT
View Forum Message <> Reply to Message

Here is a possible implementation, a bit inspired from Qt, but in essence it's different than QStyle because it doesn't draw entire widgets as qt does, but parts and these parts will be drawn to ChStyle structure and Upp Controls are free to use them as wanted.

```
enum Control {
    PushButton,
    RadioButton,
    DropList,
    ...
};

enum State {
    Normal,
    Default,
    Pressed,
    Focused,
    Hot, // MouseOver
    Disabled
};

enum SysInfo
{
    HaveAnimations,
    AnimationTime,
```

```
    Composited,
    //...
};


enum PixelMetric
{
    PushButtonMargin,
    PushButtonDefaultHeight, // MacOSX...
    //...

};

class Camo
{
public:
    void drawControl(Draw& w, Control c , State s, Rect bounds, Rect clip);
    int pixelMetric(PixelMetric pm);
    int getInfo(SysInfo);
    Color getColor(Control c, State s, ... );
    // ....
    // ....
};

// CamoWin.cpp , CamoGtk.cpp, CamoMacOS.cpp, CamoUpp.cpp ,etc.
```

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by mirek on Thu, 12 Nov 2009 20:50:29 GMT
View Forum Message <> Reply to Message

IMO, there is maybe a minor problem with this not being really extensible = the set is fixed.

Also, you will have hard time pressing all specific features of those many different widgets into single interface.

Mirek

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Thu, 12 Nov 2009 22:01:22 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Maybe you have another idea of implementation, more extensible?
Anyway i have to do this or something else related to cross-platform GUI. I know i will have hard times doing that because even Chameleon after some years doesn't get things right.

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by mirek on Fri, 13 Nov 2009 16:26:07 GMT
View Forum Message <> Reply to Message

Well, I am sorry, but I am confused about the thesis.

For a moment, I thought you are going to do Chameleon replacement (for U++?).

Anyway

Quote:
I know i will have hard times doing that because even Chameleon after some years doesn't get things right.

This in cases is only because it is hard to interpret all corners of host platform behavior, especially when one is not designed to be interpreted (like Gtk).

Besides that, I am very happy with Ch design, especially its defaulting/cascading ability... - I mean, you can alter only a minimum parameters and the thing still works. Also, I am very happy about Ch being completely parametric, you only are changing values of styles.

Mirek

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Fri, 13 Nov 2009 19:34:24 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 13 November 2009 18:26Well, I am sorry, but I am confused about the thesis.

For a moment, I thought you are going to do Chameleon replacement (for U++?).

Well I'm not going to replace entire Chameleon, I'm thinking only at the part that read the look from the system. If U++ get ported to MacOSX how will the glue code which read the look from system will looks like? I'm trying to do a lib with which the controls will interact in the same way and system dependent parts to be hidden behind it. Well maybe you know but MacOSX have specifications about the space between buttons, the distance from the outer frame of the window, etc and if an application doesn't respect these hints it will look "ugly" compared to other Mac applications and i know that mac users are choosy when talking about the look. So at this time

U++ Chameleon can't do these kind of actions like reading size hints and what I'm trying to do is library to make it possible.
If you believe it doesn't worth then I will make something else regarding my thesis. When I've choose cross-platform GUI as a subject of my thesis I've thought that I will may help U++ while working on thesis and also do something i like to do.

You're the one who have the final word It worth or not?
(Still some changes to widgets will have to be made in order to use the new system for size hints and how widgets will behave)
If you think that is simple and better to fix the current Chameleon state then i apologize for wasted your time.

Andrei

P.S. I may send a list with current glitches in Chameleon in gtk+ or better try to solve these and send the patches.

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Sun, 15 Nov 2009 21:04:50 GMT
View Forum Message <> Reply to Message

http://www.ultimatepp.org/forum/index.php?t=msg&th=4866& amp;start=0&

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 919 times

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by mirek on Thu, 11 Feb 2010 18:12:26 GMT
View Forum Message <> Reply to Message

Hm, I propose you to check existing Styles of CtrlLib first.

Start with ScrollBar or MultiButton...

I would say the problem of Chameleon is not anything that could be solved by proposed API. The problem is how to get required information from host-OS.

Plus, such fixed API is somewhat too rigid. Note that Style system is naturaly 'cascading' and easily extensible, everything has a nice default etc... I can say "this button is supposed to look like normal button if not said otherwise" etc....

In fact, I was thinking about something completely different. What I would like to see is that any widget somehow fits with target platform, including custom widgets you develop.

Instead of providing drawing api, I would perhaps tried to "classify" host GUI, like buttons are rectangular, slightly rounded, extremely rounded, base dialog background, base button face, platicity etc.. Having these colors and visual features, you could easily choose the style of any widget, even if you do not have enough info from host os.

Mirek

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon

luzr wrote on Thu, 11 February 2010 20:12Hm, I propose you to check existing Styles of CtrlLib first.

Start with ScrollBar or MultiButton...

I would say the problem of Chameleon is not anything that could be solved by proposed API. The problem is how to get required information from host-OS.

That's what my API is trying to solve, how to get required information from OS and present it to U++ in a portable way.
Checking existing Styles would help me to decompose controls in smaller parts which would get returned by my API if they are requested by U++.

Quote:
Plus, such fixed API is somewhat too rigid. Note that Style system is naturaly 'cascading' and easily extensible, everything has a nice default etc... I can say "this button is supposed to look like normal button if not said otherwise" etc....

I will keep Style in every Ctrl, i will not even modify Style structure if it is complete (having all controls states), so 'cascading' of styles will be preserved. My API only action as a replacement of ChWin32.cpp and ChGtk.cpp - first part of these files which are responsible for getting required information from the OS.
What my API is trying to solve is hiding complexity of getting information from OS and present to U++ in a portable manner. For example what my API is trying to do is to hide this:

```
int efp = EP_EDITTEXT;
int efs = 1;
int ebsx = max(2, XpInt(XP_EDIT, efp, efs, 2403/*TMT_BORDERSIZE*/));
int ebsy = max(1, XpInt(XP_EDIT, efp, efs, 2403/*TMT_BORDERSIZE*/));
Image ee = XpImage(XP_EDIT, efp, efs, SColorFace(), Size(10 * ebsx, 10 * ebsx));
ImageBuffer eb(ee);
eb.SetHotSpot(Point(ebsx, ebsy));
ee = eb;
EditField::Style& s = EditField::StyleDefault().Write();
s.activeedge = false;
s.edge[0] = ee;
```

into something like this:

```
unsigned char buffer[12*12*4]; // => imagecx * imagecy * sizeof(RGBA)
Point p1, p2;
CamoGetImage(/* unsigned char */ &buffer, EditText, State_Normal, 12, 12, &p1, &p2);
ImageBuffer ib = RGBAToImageBuffer(buffer);
ib.SetHotSpot(p1);
Image ee = ib;
EditField::Style& s = EditField::StyleDefault().Write();
s.activeedge = false;
s.edge[0] = ee;
```

I've only changed how things are get from OS, the real get will be in my dll/so implementation.

Quote:
In fact, I was thinking about something completely different. What I would like to see is that any widget somehow fits with target platform, including custom widgets you develop.

Custom widgets will fit in platform if the programmer use available colors provided by OS (through Camo interface - somehow like GetThemeColor).
Quote:
Instead of providing drawing api, I would perhaps tried to "classify" host GUI, like buttons are rectangular, slightly rounded, extremely rounded, base dialog background, base button face, platicity etc.. Having these colors and visual features, you could easily choose the style of any widget, even if you do not have enough info from host os.

Mirek
IMO having something like that will fail to provide the same look and feel as the OS have. Take for example MacOSX Aqua theme, it's animated, buttons have a blue "wave" on them.
It's possible to provide also an API which can tell if the button is rectangular, a bit rounded, extremely rounded, that would be a plus but not the base for widgets.
Why then not use our own widgets without taking the look from OS take only colors and colorize widgets based on that? I think that simple wouldn't fit.

If you'll have to start from scratch developing a GUI toolkit which way you'll choose - current way of U++ and Qt of drawing every widget or wxWidgets way of being a wrapper over OS API? Drawing widgets have the advantage that you may customize them as you want while using a wrapper have the advantage that your application will look and feel exactly as other native applications.

Andrei

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by mirek on Sat, 13 Feb 2010 12:26:12 GMT
View Forum Message <> Reply to Message

[quote title=andrei_natanael wrote on Thu, 11 February 2010 14:11]

unsigned char buffer[12*12*4]; // => imagecx * imagecy * sizeof(RGBA)
Point p1, p2;
CamoGetImage(/* unsigned char */ &buffer, EditText, State_Normal, 12, 12, &p1, &p2);

I still do not quite see the point. So, you have moved implementation to CamoGetImage, OK. This adds a little code here, but can help a bit.

BUT notice that even the organization of code that you consider "common" can be drastically different for XP and Gtk. I am afraid that at some point, you will end with something like:

```
if(CamoIsPlatform(GTK)) {
....
}
else {
....
}
```

which will be only a bit worse than today...

That said, I believe that the Ch*.cpp code should get some significant overhaul. And I can still be very wrong about Camo...

Mirek

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by mirek on Sat, 13 Feb 2010 12:31:30 GMT
View Forum Message <> Reply to Message

And one more detail about Ch difficulities. The main problem in Gtk is that when the styling engine renders the widget, it has the pointer to the real widget.

Styling engine then can alter behaviour based on the settings of the widget, and they DO, including such subtle things like "button is now child of menubar". So in order to get something meaningful, you have to emulate all these things, create all hierarchies and hope for the best (that is, that another engine will not do something you have not accounted for yet).

Mirek

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by Tom1 on Sat, 13 Feb 2010 18:19:07 GMT
View Forum Message <> Reply to Message

Hi,

Please, never add any dependencies to any SOs/DLLs in packages belonging to uppsrc. In other 'optional' nests, I don't care, but just keep the uppsrc clean of them.

I seriously think the basic GUI framework should be kept as simple and robust as possible.

Ultimate++ has become a vitally important tool for me to create commercial software products and anything that makes things more complex, usually causes loads of customer support that can't be charged for.

I can understand that 'eye candy' is needed for certain segment of software products, but simplicity and reliability is more important for me and my clients, so I suggest the 'eye candy' packages come in only as an option for those interested -- not by default.

Kind regards,

Tom

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Sat, 13 Feb 2010 19:30:02 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 13 February 2010 20:19Hi,

Please, never add any dependencies to any SOs/DLLs in packages belonging to uppsrc. In other 'optional' nests, I don't care, but just keep the uppsrc clean of them.

I seriously think the basic GUI framework should be kept as simple and robust as possible. Ultimate++ has become a vitally important tool for me to create commercial software products and anything that makes things more complex, usually causes loads of customer support that can't be charged for.

I can understand that 'eye candy' is needed for certain segment of software products, but simplicity and reliability is more important for me and my clients, so I suggest the 'eye candy' packages come in only as an option for those interested -- not by default.

Kind regards,

Tom

Hello Tom,
So the solution would be one exe for KDE (if I'll implement KDE look using KDE libs), one for GTK+, because right now gtk+ is statically linked in U++ and one exe if i don't care about platform integration (X11 only with flagNOGTK). I know it may looks more robust but I'll have to provide 3 different executables for (i.e.) same client using Gnome, KDE or other desktop managers sometime. My solution was to avoid statical linkage and provide only an exe which would work in all 3 cases. If user is running Gnome then it will use gtk+ (indirectly through camo-gtk.so), in case that the same user is signing off from Gnome and start his KDE desktop environment then the same exe will get the look for system using camo-qt.so. If the user isn't using a D.M. at all or is using a DM which doesn't use either gtk+ or qt then the application will use U++ look. Existence of gtk or qt will not be a problem for user like it is now (with gtk static link). Is not a problem if my

program doesn't find at runtime one of those two libs (camo-gtk/qt.so) because it will use U++ look but now if we take an U++ program which need gtk+ and move the exe in KDE it will fail to start if gtk libs are not installed. I see the robustness more in my cases than in currently U++ state which force the developer to specify which libraries will be available on system.
Perhaps an pseudo-code of implementation will make my design approach more clear;

```
if (CurentRunningDesktopManagerIsGnome) {
  ChHostSkin will make use of camo-gtk.so
} else {
  if (CurrentRunningDesktopManagerIsKDE) {
    ChHostSkin will make use of camo-qt.so
  } else {
    ChHostSkin will use U++ style (ChStdSkin)
    // falling back to U++ style if qt/gtk aren't installed
  }
}
// BTW, this kind of check is running once at application start-up so there will be no penalties using an "external" so/dll
// And because camo-qt and camo-gtk share the same interface ChHostStyle of both would look identical, in fact it will be only one ChHostSkin implementation
```

camo-gtk/qt.so are required for portability, if there is another possibility to make use of gtk and qt in the same application without using an "external" so/dll then let me know. IMO it's not possible that because you have to link with qt at compile time because qt lib are C++ libs and you cannot use dlopen/dlsym with them.

Andrei

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Sat, 13 Feb 2010 19:33:37 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 13 February 2010 14:26
I still do not quite see the point. So, you have moved implementation to CamoGetImage, OK. This adds a little code here, but can help a bit.

I think i have to come with a real implementation not just mock-ups to make the situation more clear.

Andrei

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by Tom1 on Sat, 13 Feb 2010 20:55:48 GMT

Hi Andrei,

Sorry, I probably did not make myself clear: I'm not judging your Camouflage project and your attempt to enhance U++. I'm not even familiar with the current Chameleon implementation -- not to mention your Camouflage plans in detail.

The point I want to make is that for those of us, who do not need widget resemblance with the currently running window manager, a single statically linked executable with the default U++ look is quite enough. The knowledge that the application will not require any SOs/DLLs on any window manager / platform is important.

I understand from your post that the SOs/DLLs would be OPTIONAL and even their 'accidental' absence would not cause any trouble for the application's ability to run on top any window manager. Is this true? If so, it's OK by me.

Finally, it is important to notice that if any implementation of platform compatible looks requires linkage to LGPL licensed components, it's better be dynamic linkage. If it is GPL, it's better leave alone completely.

Kind regards,

Tom

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by mirek on Mon, 15 Feb 2010 09:47:26 GMT

Tom1 wrote on Sat, 13 February 2010 13:19Hi,

Please, never add any dependencies to any SOs/DLLs in packages belonging to uppsrc. In other 'optional' nests, I don't care, but just keep the uppsrc clean of them.

I seriously think the basic GUI framework should be kept as simple and robust as possible. Ultimate++ has become a vitally important tool for me to create commercial software products and anything that makes things more complex, usually causes loads of customer support that can't be charged for.

I can understand that 'eye candy' is needed for certain segment of software products, but simplicity and reliability is more important for me and my clients, so I suggest the 'eye candy' packages come in only as an option for those interested -- not by default.

No worry, your concerns are absolutely shared - I have the same issues to work with - supporting apps that consist of many dynamic libraries is significantly more difficult.

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by andrei_natanael on Mon, 15 Feb 2010 10:39:36 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 15 February 2010 11:47Tom1 wrote on Sat, 13 February 2010 13:19Hi,

Please, never add any dependencies to any SOs/DLLs in packages belonging to uppsrc. In other 'optional' nests, I don't care, but just keep the uppsrc clean of them.

I seriously think the basic GUI framework should be kept as simple and robust as possible. Ultimate++ has become a vitally important tool for me to create commercial software products and anything that makes things more complex, usually causes loads of customer support that can't be charged for.

I can understand that 'eye candy' is needed for certain segment of software products, but simplicity and reliability is more important for me and my clients, so I suggest the 'eye candy' packages come in only as an option for those interested -- not by default.

No worry, your concerns are absolutely shared - I have the same issues to work with - supporting apps that consist of many dynamic libraries is significantly more difficult.

There should be no worry
I found a way to use it in U++ without modifying U++ source, someone would just add Camo package to his project and then Camouflage will be used instead of U++ theming - that's easy because current skinning implementation .

Andrei

Subject: Re: Graduation thesis - Camouflage - a replacement of(a part of) Chameleon
Posted by mirek on Mon, 15 Feb 2010 11:18:28 GMT
View Forum Message <> Reply to Message

andrei_natanael wrote on Mon, 15 February 2010 05:39luzr wrote on Mon, 15 February 2010 11:47Tom1 wrote on Sat, 13 February 2010 13:19Hi,

Please, never add any dependencies to any SOs/DLLs in packages belonging to uppsrc. In other

'optional' nests, I don't care, but just keep the uppsrc clean of them.

I seriously think the basic GUI framework should be kept as simple and robust as possible. Ultimate++ has become a vitally important tool for me to create commercial software products and anything that makes things more complex, usually causes loads of customer support that can't be charged for.

I can understand that 'eye candy' is needed for certain segment of software products, but simplicity and reliability is more important for me and my clients, so I suggest the 'eye candy' packages come in only as an option for those interested -- not by default.

No worry, your concerns are absolutely shared - I have the same issues to work with - supporting apps that consist of many dynamic libraries is significantly more difficult.

There should be no worry
I found a way to use it in U++ without modifying U++ source, someone would just add Camo package to his project and then Camouflage will be used instead of U++ theming - that's easy because current skinning implementation .

Andrei

Actually, it is true that you can provide nice .dll based skinning without even changing a single line in CtrlCore/CtrlLib...

Mirek
