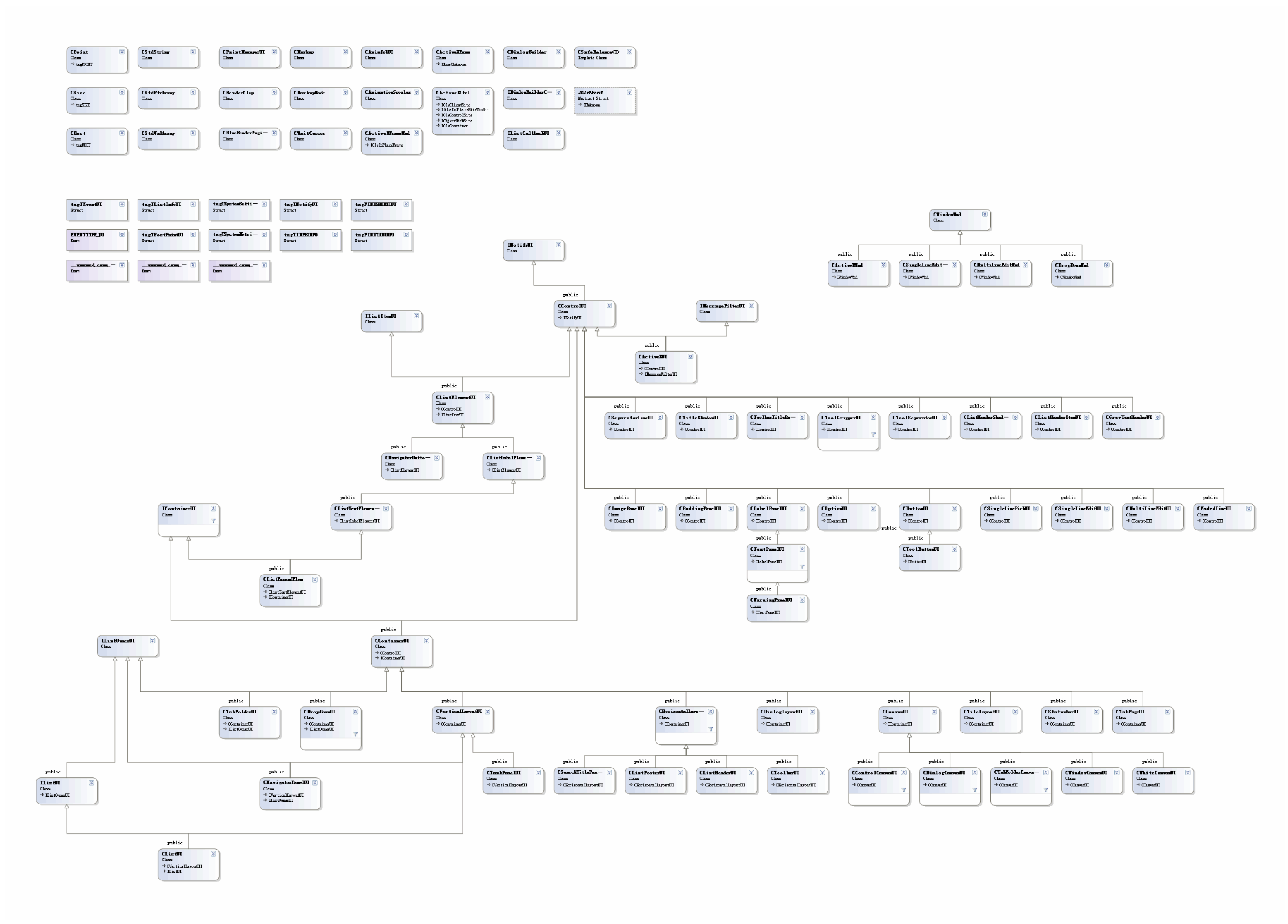
开源 DirectUI 类库介绍及类图分析

2011-8-25

丹麦作者 bjarke（比亚克）的开源 DirectUI 类库 UIlib 类图

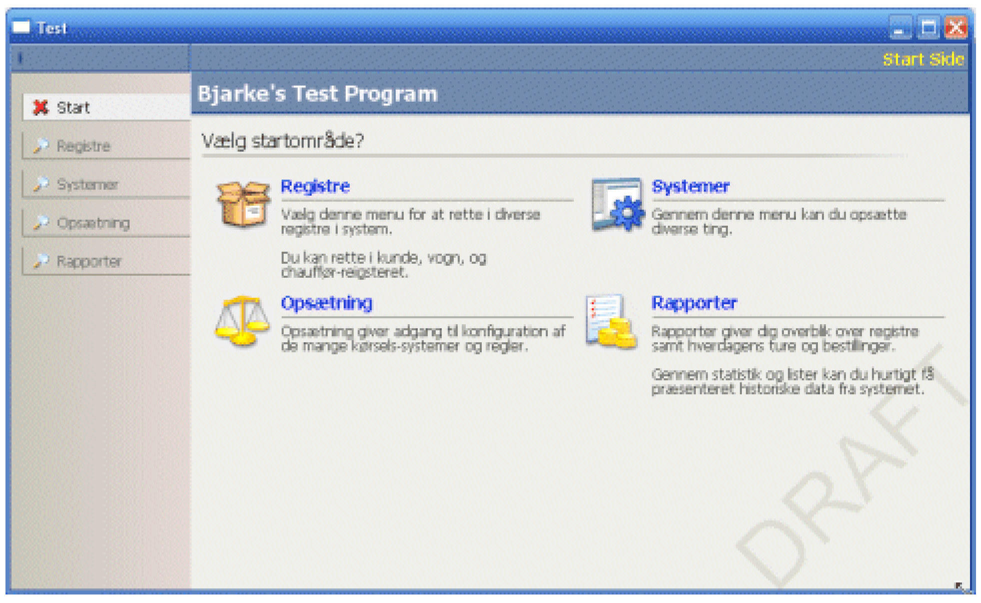
源码及相关资料参见 http://www.viksoe.dk/code/index.htm
类图分析绘制：borlittle
请放大后查看



另附上 bjarke 的一篇关于 DirectUI 的经典文章《无窗口的 UI》

UI: Become windowless

http://www.viksoe.dk/code/windowless1.htm
原作者：bjarke（比亚克）
国籍：丹麦
翻译：borlittle



Bjarke's Test Program

Vælg startområde?

Registre
Vælg denne menu for at rette i diverse registre i system.
Du kan rette i kunde, vogn, og chauffør-reigsteret.

Systemer
Gennem denne menu kan du opsætte diverse ting.

Opsætning
Opsætning giver adgang til konfiguration af de mange kørsels-systemer og regler.

Rapporter
Rapporter giver dig overblik over registre samt hverdagens ture og bestillinger.
Gennem statistik og lister kan du hurtigt få præsenteret historiske data fra systemet.

A request on the Yahoo WTL newsgroup and a blog-article sparked some interest to look closer at creating a **windowless user-interface**. Traditionally, Windows applications are built upon the GDI/User windowing hierarchy, and thus restricted in several areas. While you can certainly generate a nice application quickly with the built-in control-set and get a standard clean look, you will soon enough stumble upon the limitations of the Windows controls - especially if you want to build something that looks a little more flashy. The native Win32 custom-draw/owner-draw technique is limited, next to no support for transparent windows, rigid control scaling/resizing and the Common Controls are merely remnants of Windows Explorer, Office or Internet Explorer widgets.

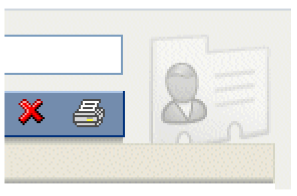
Yahoo WTL 新闻小组上的一个要求和一个博客闪现出了一些对于创建一个无窗口用户接口的兴趣。传统意义上，窗口应用时构建在 GDI/User 窗口体系之上的，因此在一些地方受到限制。当你能够使用内建的控件集快速的生成一个好看的应用以及得到一个标准干净的外观时你就会深刻感受到 Windows 控件的局限带来的困难，尤其是你想要构建一些看起来有些华丽的界面。之前的 Win32 custom-draw/owner-draw 技术有许多限制，如几乎不支持透明窗口，生硬的控件缩放和调整，这些通用控件如今基本上只保留作 Windows Explorer, Office or Internet Explorer 上的小部件了。

A windowless user-interface doesn't make use of the native Win32 controls. Instead it custom paints every control and widget on the screen - including labels, buttons, list control, etc. etc. Variations of this kind of interface is not new: "Skinning" has been widely accepted for its cool application-look, as seen in Nullsoft's WinAmp and the MS Media Player. But skinning is usually only feasible for small applications hosting a limited number of dialogs. If you're building a large database-driven application, you'll want to have a solid framework to back your dialog design up.

一个无窗口的用户接口 UI 不会使用传统的 Win32 控件。而是在屏幕上自绘所有的控件和部件，包括标签、按钮、列表控件等等。大量的这种接口并不新奇：皮肤技术作为绚丽的应用外观已经被广泛的接受，如 Nullsoft's WinAmp 和 MS Media Player。但是皮肤通常只是在小型的只包含有限数量的对话框应用中。如果你要构建一个大型的数据库驱动应用，你会需要一个固定一致的框架来支撑你的对话框设计。

So I was looking at the Microsoft CRM application. In my opinion, this is one of the most successful DHTML interfaces I've seen (I should know because I've previously been assigned on a large browser-based CRM product). The MS CRM interface looks very Windows XP-like, but it also has several cool features, such as a navigation bar, a highly flexible list control and several shaded/gradient tool panels. I thought this would be a nice look to build for my windowless framework. I've previously tested a DHTML interface with this look, but wasn't satisfied - mostly because integrating with the browser (IE) was painfully complicated.

所以我一直在关注 Microsoft CRM（微软的客户关系管理）应用。我认为这是我看的最为成功的 DHTML 接口之一（我应该知道的，因为我之前已经分配到一个大型的基于浏览器的 CRM 产品。MS CRM 界面看起来非常有 Windows XP 风格，但是它也有一些很酷的特征，比如一个导航条、一个高度浮动的列表控件以及一些阴影/灰度工具面板。我想这应该会是我的无窗口框架很好的外表。我之前已经测试了一个这种界面的 DHTML 接口，但是并不满意，主要是因为和浏览器 IE 结合是一件非常痛苦的事。



Even if constructing a windowless user-interface is a lot of hard work, it pays off in the end. But you do need to implement your own dialog builder, button-design, keyboard interface and lots of little things you take for granted. Actually it's not the painting of buttons and toolbars that take the most planning and effort; it's supporting keyboard shortcuts, tab-navigation, automatic tooltips and a scalable design. And then there are the mandatory - but often overlooked - features, such as list column-sorting, intuitive focus changes and overflow scrolling that need to be addressed.

尽管构建一个无窗口的用户接口 UI 是一件非常难的工作，最终是值得这样做的。但是你的确需要实现你自己的对话框构建器，按钮设计，键盘接口和许多你认为需要的东西。实际上绘制按钮和工具条并不是最费考量和努力的而是支持快捷键、tab 导航、自动提示和可扩展的设计。然后是托管—但经常被忽略的特征，如列表列排序、直觉焦点切换和卷动溢出需要重点考虑的。

The design 设计

My **Direct UI** user-interface takes bits and pieces from other UI models.
我的 Direct UI 用户接口或多或少借用了其他 UI 模型。
Specifically it borrows ideas of the "controls-inside-controls" design from the Java AWT framework, XML dialog-resources from .NET, event-bubbling from DHTML and C++ helper classes from MFC. The code doesn't actually use the WTL library like most of my other user interface samples on this website. It's pure Win32 calls.
尤其是它借用了 JAVA AWT 框架"控件中的控件"设计思想，.NET 的 XML 对话框资源，DHTML 的事件起泡和 MFC 的 C++帮助类。这些代码实际上并没有像其他 UI 例子一样使用了 WTL 库。它是纯粹的 Win32 调用。
One of the requirements for the framework was that it could be placed in an external DLL. This had a *nasty impact on the design*, because C++ templates are not by any sane method exportable in a DLL. So, a clean C++ old-skool (MFC) design is enforced, with single inheritance and few (if not none) macros.
对控件的一个要求是它必须放置在一个扩展的 DLL 中。这对设计有负面的影响，因为 C++模板在任何健全 DLL 中不能被导出。因此一个干净的 C++守旧设计(MFC)通过简单的继承和一些宏定义是被迫的。
A window is built by nesting controls inside each other. Some controls are containers (such as the ToolBar, which contains ToolButtons). To insert a ToolBar button, you simply construct a `CToolButton` C++ class and add it to the `CToolPanel` control container. Most containers define layout algorithms such as the vertical layout-panel, which arranges the contained elements below each other - a feature known from Java AWT.

一个窗口必须由互相嵌套的控件构成。另一些控件是容器，如工具条包含了工具按钮。为了插入工具条按钮，你只需简单的构建一个 CToolButton C++类然后将其添加到 CToolPanel 控件容器，它会安排包含的元素在底层-这个特征来源于 Java AWT。

Using a Java-like design, such as the layout containers, make sure that your controls will rescale automatically when the window is resized. But anyone who has done any Java development knows how limiting its control layout features are and has wished to kick the authors of the GridBagLayout class in the nuts. So there's also room for a dialog layout-panel, which allows you to put controls at a fixed position, but with the option of scaling based on various stretching rules.
使用一个像 JAVA 的设计，诸如层容器，一定要保证当窗口大小改变时你的控件是能够自动的调节的。但是任何一个做过一些 JAVA 开

发的都知道它的层控件特征有多么局限而且很想踢 nuts 中 GridBagLayout 类的作者。因而对于一个对话框层面板有空间允许你把控件放在一个固定的位置，但是使用尺度变换基于变化的拉伸规则。
Constructing an entire window by hand is tedious, so there's a small XML parser included (extremely fast, extremely non-compliant) which parses and builds a window from an XML string.
手动的构建一个完整的窗口是非常枯燥的，因而需要包含一个小的 XML 解析器（相当的快）用于从一个 XML 字符串中解析和构建一个窗口。
The framework caches all of its Win32 brushes and pens. Most of the GDI resources saved on window handles are probably spent on this. But these kinds of objects are light-weight; it's the paint job that gets an incredible overhaul.
框架隐藏了所有的 Win32 刷子和画笔。大多数的节省的窗口句柄的 GDI 资源可以用到这上面。但是这些对象是轻量级的，但是绘画的工作需要严格的检查。
Native Win32 controls are used only for the `EDIT` control. This control contains so much functionality that it would take ages to do a decent replacement. Single-line edit controls are created on the fly (when you click on the frame) and multi-line edits are always visible. So the framework does have the ability to embed native Win32 controls and even ActiveX controls, but at the expense of screen flickering and severe restrictions in the visual effects I'm planning.
传统的 Win32 控件只用于 EDIT 编辑控件。这个控件包含了许多的可能需要很多时间才能完成像样替换的功能特性。单行编辑控件在 fly（当你点击框架时）时创建而多行编辑框总是可见的。所以框架也有能力去嵌入传统的 Win32 控件甚至 ActiveX 控件，但是我主要考虑的是屏幕的闪烁和视觉上的严格约束。

Alluring text 吸引人眼球的文字

The real benefits of this type of interface are the simple things. For instance, you can with one single swoop replace all occurences of Win32 DrawText() with DrawPrettyHtmlFormattedText() - which instantly gives you icons, customizable text-colors and clickable hyperlinks in the entire user-interface. With one single line of code, you can now add HTML links in the statusbar panel.
这种接口的实际好处是很简单的事。比如你可以用一个简单的 DrawPrettyHtmlFormattedText()替换所有出现的 Win32 DrawText()函数，这快速的使你的图标、用户定制的文字颜色和可点击的超链接为完整的 UI。使用一行代码，你现在可以添加 HTML 链接到状态条面板。
Similar, you are not restricted to only put CToolButton controls inside the ToolBar. Any kind of control can be added. This also goes for the list control, which quickly can be made cool by adding some group-labels, or just by adding buttons or HTML links. Since the individual controls rarely erase the background, most of them will actually fit transparently inside the other container controls, so once you've made a neat widget it can be reused in the entire user interface.
类似的，你不用局限于只把 CToolButton 控件放到 ToolBar 中。任何类型的控件都可以添加。这也适用于列表控件，这使得可以快速的添加一些群标签或者只是添加一些按钮或者 HTML 链接而变得酷起来。由于独立的控件很少擦除背景，它们大多数会实际上适合在别的容器控件中变得透明。所以一旦你已经生成了一个灵巧的部件它就可以被 UI 完全的重用。

3D Animations3D 动画

Another cool effect that can be easily added is the ability to do 3D rendering directly on the canvas. Just throw in some DirectX calls and you'll be able to write cool page transitions, highlight a control with some cool sparkling particle-effect or just put some glow effect on the edit-box with focus.
You can also read about how I integra ted DirectX into the application.
另外一个很酷的效果是可以轻易的添加这个性能来直接在画布上实现 3D 渲染。只需要扔一些 DirectX 调用你就可以写出很酷的页面渐变、用一些很酷的发光粒子效果高亮一个控件或者当一个编辑框获得焦点时添加一些发光效果。你可以阅读一些我如何集成 DirectX 到应用中。

The sample 例子

This may all sound very nice. However, the sample presented here is just the initial stage of a windowless framework I'll use for further testing. The library will need to mature a bit and go through a few rewrite cycles before I'll attempt to actually incorporate it in a real application. Still, it's pretty functional already. Enjoy.
这可能听起来非常不错。但是，这里提供的示例只是我以后要用于更多测试无窗口框架的初步阶段。库将会更成熟一些而且在我尝试实际结合到实际的应用之前会经过一些重写周期。尽管如此，它也工作得非常好了。尽情享受吧。
Wangchyz has a project on Google Code where's he's trying to extend the library, adding fixes and new samples.
Wangchyz 已经在 Google Code 上有一个项目，他正在尝试扩展这个库，添加和修正一些新的例子。
The sample should be run with **DirectX 9** installed and a modern 3D graphics card. Otherwise you will not be able to view the pretty 3D animations.
例子应该在装了 DirectX9 和一个新一些的 3D 图形卡的系统上。否则你可能看不到漂亮的 3D 动画。

Source Code Dependencies 源代码支持

Microsoft Visual C++ 6.0

Microsoft DirectX 9 SDK

Useful Links 有用的链接

An extended version: Duilib

Download Files 下载文件

[Demonstration (92 Kb)](#)
[Source Code (127 Kb)](#)

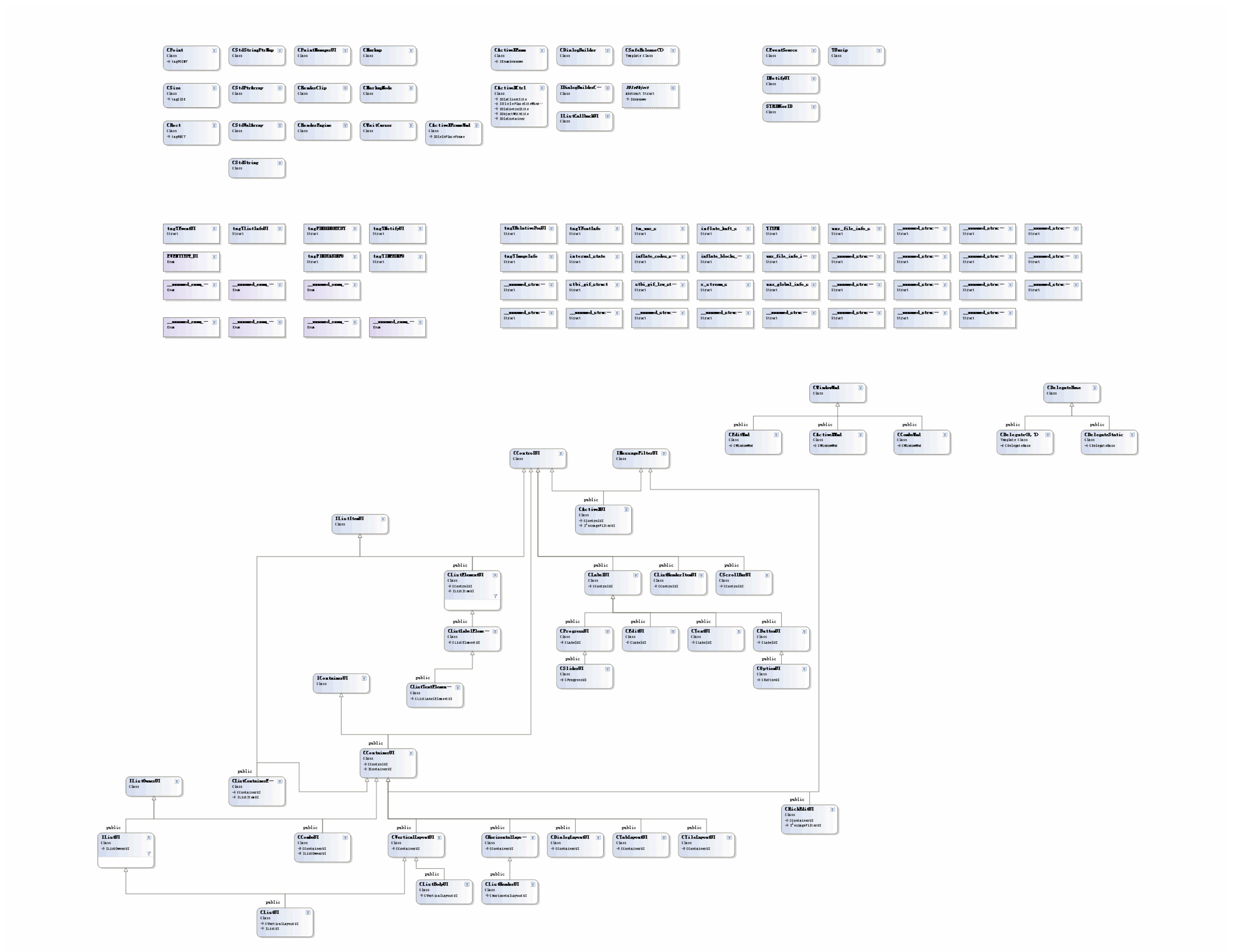
关于 DirectUI 的更多文章和资料，可以参考
http://www.taxue.org/
http://my.oschina.net/achellies/blog/17654
http://www.cppblog.com/sleepwom/archive/2010/08/07/122554.html
http://blog.csdn.net/jiangsheng/article/details/5404320
http://blog.csdn.net/wangchyz/article/details/5818669

国内开源 Duilib 项目类库类图

源码及相关资料参见 http://code.google.com/p/duilib/
类图分析绘制：borlittle
请放大后查看

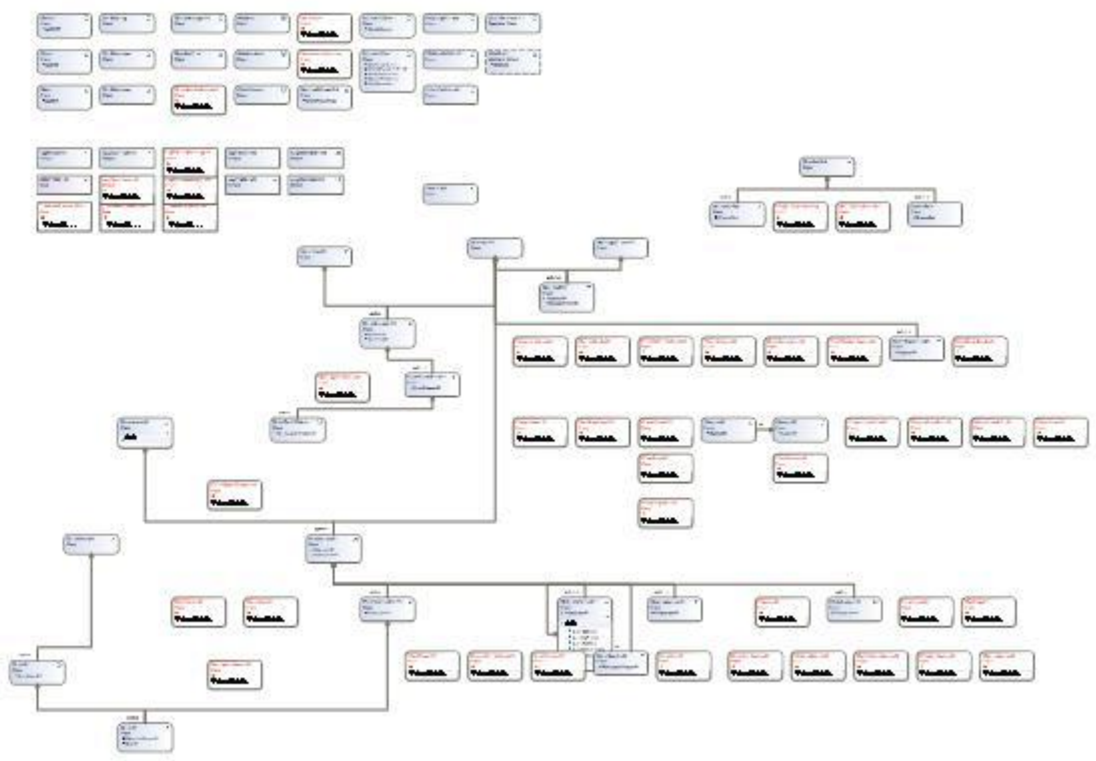
Duilib V1.1

如图所示：



DuilibV1-1 类图与 UIlib 的改变

如图所示：



上图中红色部分即为已经更改了的类和结构，可参照 UIlib 类图和 Duilib 类图。