# 简单说明

TestLib是ATL用的例子，TestMFC是MFC用的例子，MakeSkin是专用的皮肤打包器，打包后是单个文件，也可以把它嵌入到资源中。例子最开始，都需要先UILib::LoadSkin加载皮肤，一般Debug下指定路径，加载路径下的图片文件，便于开发调试，Release下指定打包后的皮肤文件或资源，提高效率，解包算法效率优于只存储的zip包。

UILib.sln编译为动态库，例子TestLib使用时：①在项目属性-通用属性-引用中，添加UILib、UIWnd两个项目的引用，用于项目依赖和链接；②在项目属性-配置属性-C/C++-预处理器-预处理器定义中，添加宏定义UILIB\_IMPORTS；③在stdafx.h中include两个头文件：..\UILib\UILib.h、..\UILib\UIDialog.h。

UILib2.sln编译为静态库，例子TestLib2使用时，只添加UILib2项目的引用，因为它包含了UIWnd的文件，也不用添加宏定义UILIB\_IMPORTS。

先从基本的介绍，CImagex是公用图像类，有2个成员shared\_ptr<CImage> m\_spImage、CRect m\_rect，分别是图像的指针、和自己在图中占的区域。比如一个4态的按钮，每态宽高为30、20，则图像总宽高为120、20，加载时指定“按钮.png:4”，就加载了第1态，用SetFrameIndex(n)切换其它态，从源码可以看到，切换只是把自己的m\_rect改了一下。

接下来是CUIView、CUIControl、CUIRootView，这3个类把整个UI库驱动起来了，其它控件只是继承CUIView或CUIControl，并特化自己的鼠标、绘制消息。主要看下这3个类的源码，看懂之后，其它控件就非常简单了。

CUIView是所有UI的基类，有一些设置、获取属性的方法和成员变量，同时它也是一个容器，有成员变量vector<CUIView \*> m\_vecChilds，和添加、删除、查找、布局子控件的方法；CUIControl继承自CUIView，从名字就知道它是控件的基类，像CUIButton、CUICheckBox等都是继承自它，这个类提供的方法很少很好懂；CUIRootView继承自CUIView，它是顶层视图，用于和Window、WindowLess交互，任何控件都能用GetRootView方法使用它。整个界面是树形结构，根是RootView，往下是各层各个View、Control。

处理鼠标消息时，通过调用OnHitTest 找出Z序最高的Control，也就是最上层控件，让它处理鼠标点击消息，即使它不处理，也不给其它控件机会（就像windows窗口的行为一样）。如果想让它对鼠标点击透明，比如CUIImage等，则设置m\_bClickable = false。

MouseEnter、MouseLeave这2个消息特殊，如果它的父视图是Control，则会通知父视图也MouseEnter、MouseLeave。比如浏览器标签：标签是按钮，右边的×号是它的子按钮，鼠标放在×号上，标签和×号同时MouseEnter了。

UI库最核心的部分，是如何布局控件，算法在CUIView::CalcRect和CUIView::RecalcLayout。对于每个View（所有控件都是View，包括Control、RootView），当需要重新布局时，先CalcRect算出自己的区域，再比较如果和之前不一样，则RecalcLayout重新布局自己的子控件；每个子控件也先算自己的区域，再重新布局它的子控件，这样依次递归。

CalcRect算法：传进来的rect，对于左右方向，有左、右、宽3个属性，都不设置的话：left=rect.left、right=rect.right。只设置1个属性的情况：只设置左则left=rect.left+左、right=rect.right，只设置右则left=rect.left、right=rect.right-右，只设置宽则left、right在rect居中。同时设置2个属性的情况：设置左、右就定下来了，设置左/右、宽就能算右/左。如果同时设置3个属性则只用左、右。上下方向同理。

指定了左、宽时，可以同时指定左剪裁，即算出自己的rect2后，剩余rect.left=rect2.right并返回，是否剪裁看SetLeft的第2个参数，xml中对应的是数字后加\*号，右、宽，上/下、高剪裁都同理。

如果控件使用了图片，比如Button、Image等，可以不用设置宽高，初始宽高就是图片的宽高，也可以根据需要SetWidth、SetHeight。

RecalcLayout算法：初始rect等于m\_rect，把它传给第1个子控件，让它按照约束，CalcRect算出自己需要的区域，并把剪裁剩下的区域返回，然后再传给第2个子控件同理计算，这样依次传给剩下的所有子控件。如果想自定义子控件的布局，则重写RecalcLayout方法，并对每个子控件SetRect。

例如构建一个简单窗口，先在RootView上用top、height划出标题栏并剪裁，剩下的就是客户区。再在标题栏View上用right划出关闭按钮并剪裁，再用right划出最小化按钮，然后用left划出标题等等。

关于RecalcLayout性能，使用了延迟布局算法，任何控件添加、删除、改变上下左右宽高，都会InvalidateLayout并记录m\_bNeedLayout，在最后需要WM\_PAINT前一次性布局所有需要的，再DoPaint绘制。

构建界面支持纯代码、xml配置、或两者结合，例子里都有，xml中每个控件可配置的属性，可以参考每个类cpp最后的OnLoad，注意还有\_\_super::OnLoad中的。

因为UI逻辑和Windows窗口是分离的，所以ATL、MFC都可以直接使用UILib动态库，然后再实现各自的CUIWindow类就行了。UIWnd静态库包含了ATL使用的CUIWindow，TestMFC项目包含了MFC使用的CUIWindow，各自都有使用例子。UILib也可以给无窗口WindowLess使用，参考CUIWindow把消息、绘制、区域传给RootView就行了。

# 各类介绍

每个类源码都非常短，可以自己看看，也可以只看头文件，会用就行，以后有时间再写。