目　录

1.1 项目介绍

该项目基于 Duilib 界面库，实现一个运行于 Wince 平台的界面库 CULib。该界面库针对嵌入式硬件的特点和 Wince 操作系统的特性对 Duilib 进行重构和优化。CULib 将用户界面和处理逻辑分离，极大地提高用户界面的开发效率。并且使用 xml 来描述界面风格和界面布局，可以很方便地构建高效、绚丽、易于扩展的界面，而且 xml 语法简单，易于学习和掌握。

在 Wince 环境下的 GUI 软件一般采用传统的 MFC 界面库，开发出的软件不美观、界面细节处理不好、使用硬编码、开发效率低下、生成程序体积大，而且传统 MFC 界面美化大都使用 HOOK 等对系统影响比较大的技术，可能会导致系统不稳定或引发其他错误。CULib 库采用 DirectUI 技术，完全基于 GDI 在窗口上自绘，弥补了 MFC 等传统界面库在 Wince 上的不足。CULib 的出现使得在 Wince 这样的嵌入式环境中构建绚丽的界面变得简单、高效。

CULib 具有以下一些技术特点：

界面与业务逻辑分离；

使用 xml 配置界面；

界面布局方式灵活多样；

支持 ZIP 和 DLL 两种资源方式；

支持 png、gif、jpg 等多种图片格式；

内置常用的控件；

支持自定义控件；

支持插件系统；

强大的事件处理机制；

类 html 字符串绘制技术；

基于 GDI 和脏矩形的高效绘制技术；

支持 alpha 混合和图片透明；

支持动态色调变换；

内存占用小。

1.2 系统整体结构

该系统由八大模块组成如下图所示，分别为：基础结构、第三方库、界面创建、绘制渲染、绘制管理、事件委托、控件集、布局管理。

基础结构：

一些比较大的工程，往往为了其可移植性或是其他的目的会编写自己的工具库，CULib也不例外。通过对基本绘图元素、字符串、数组、基本窗口的封装来辅助其他系统模块的构建。

第三方库：

CMarkup 是 xml 解析器，把内存或文件中的字符串解析为程序可识别的文档对象，供界面创建模块创建界面。XUnzip 是解压缩库，有时程序所使用的资源不是在一个文件夹下，而是被打包到一个压缩文件中，这时就需要 Xunzip 解压，使得程序使用被压缩的资源。Stb_image 用来加载不同格式的图片，被绘制渲染模块所使用。

界面创建：

程序界面不是在编译期创建的，而是在运行时由界面创建模块动态创建。当程序创建主窗口时会调用该模块以实现界面的创建。界面创建模块首先调用 XUnzip 解压缩库把资源文件减压缩或从磁盘文件读取资源，再调用 CMarkup 解析器对配置文件进行解析，解析完成后动态创建控件并设置其相关属性。最后由绘制管理进行界面绘制。

绘制渲染：

每个控件在界面上的显示都是控件调用绘制渲染模块来完成的，当收到 WM_PAINT 消息后绘制管理者会通知每个控件去调用绘制渲染模块来完成自身的绘制。

绘制管理：

绘制管理者是界面库中最重要的一个类，它不仅管理界面的绘制，还对事件进行管理。它通过与其他模块的交互来完成界面的绘制与事件分发。大部分 windows 消息都是在绘制管理模块内处理或分发的。该模块把 windows 消息转换为界面库内部的事件源，并传递给对应的控件。

事件委托：

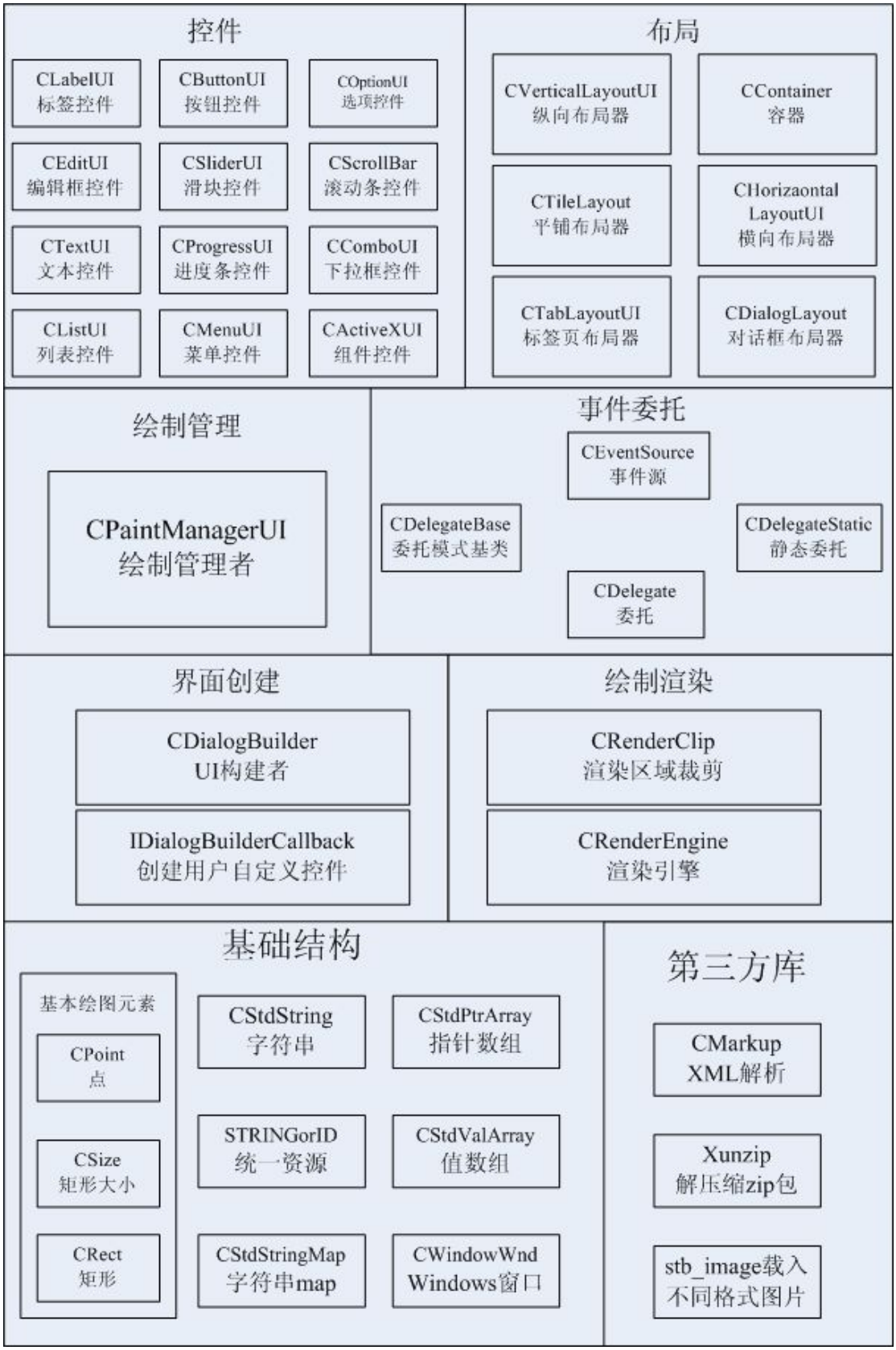
用户程序与界面库的交互有两种方式，一种是用户窗口继承监听接口，界面库向用户程序发送事件通知；另一种就是使用委托。用户使用委托可以把对控件的事件处理程序和控件的相应事件绑定。

控件：

界面库的主要作用就是方便用户编写软件界面，这当然离不开软件界面的基本元素控件。由于基于 DirectUI 技术的控件没有句柄，所以控件相对于 windows 系统是不可见的，不能像传统的界面编程一样去使用各种 windows API。这就需要界面库根据控件的特点来模拟传统控件，并把控件对应的事件发送给用户。

布局：

复杂一点的程序可能有比较多的控件，如果只依靠控件自身的属性来设置其在界面中的表现，工作量是非常大的。可以借助于布局管理器来实现控件在界面中的表现，有横向、纵向、平铺等多种布局管理器。


控件
CLabelUI
标签控件
CButtonUI
按钮控件
COptionUI
选项控件
CEditUI
编辑框控件
CSliderUI
滑块控件
CScrollBar
滚动条控件
CTextUI
文本控件
CProgressUI
进度条控件
CComboUI
下拉框控件
CListUI
列表控件
CMenuUI
菜单控件
CActiveXUI
组件控件
布局
CVerticalLayoutUI
纵向布局器
CContainer
容器
CTileLayout
平铺布局器
CHorizaontal
LayoutUI
横向布局器
CTabLayoutUI
标签页布局器
CDialogLayout
对话框布局器
绘制管理
CPaintManagerUI
绘制管理者
事件委托
CEventSource
事件源
CDelegateBase
委托模式基类
CDelegateStatic
静态委托
CDelegate
委托
界面创建
CDialogBuilder
UI构建者
IDialogBuilderCallback
创建用户自定义控件
绘制渲染
CRenderClip
渲染区域裁剪
CRenderEngine
渲染引擎
基础结构
基本绘图元素
CPoint
点
CSize
矩形大小
CRect
矩形
CStdString
字符串
CStdPtrArray
指针数组
STRINGorID
统一资源
CStdValArray
值数组
CStdStringMap
字符串map
CWindowWnd
Windows窗口
第三方库
CMarkup
XML解析
Xunzip
解压缩zip包
stb_image载入
不同格式图片

1.3 系统流程

1. 程序基本流程

-> WinMain(入口函数)

-> CPaintManagerUI::SetResourceInstance (实例句柄与渲染类关联)

-> ::CoInitialize (初始化 COM 库, 为加载 COM 库提供支持)

-> new FrameWnd (创建窗口类)

-> pFrame->Create (注册窗口类与创建窗口)

-> RegisterSuperclass (注册一个超类 即已有一个窗口类的基上再注册一个窗口类)

-> RegisterWindowClass (注册窗口类)

-> ::CreateWindowEx (创建窗口，此时触发 WM_CREATE 消息)

-> HandleMessage (WM_CREATE 消息处理 OnCreate)

-> pFrame->CenterWindow (窗口居中显示)

-> CPaintManagerUI::MessageLoop (处理消息循环)

-> ::CoUninitialize(); (退出程序并释放 COM 库)

2. WM_CREATE 的消息处理函数 OnCreate 实现：

-> FrameWnd:: OnCreate

-> m_pm.Init(m_hWnd) (主窗口类与窗口句柄关联)

-> CControlUI* pRoot = builder.Create (加载 XML 并动态创建界面无素)

-> m_pm.AttachDialog (附加控件数据到 HASH 表中)

-> InitControls (初始化控件)

-> FindControl

-> __FindControlFromNameHash

->pManager->m_mNameHash.Insert（把控件插入到 Hash 中）

-> m_pm.AddNotifier 增加通知处理

3. CDialogBuilder::Create 实现

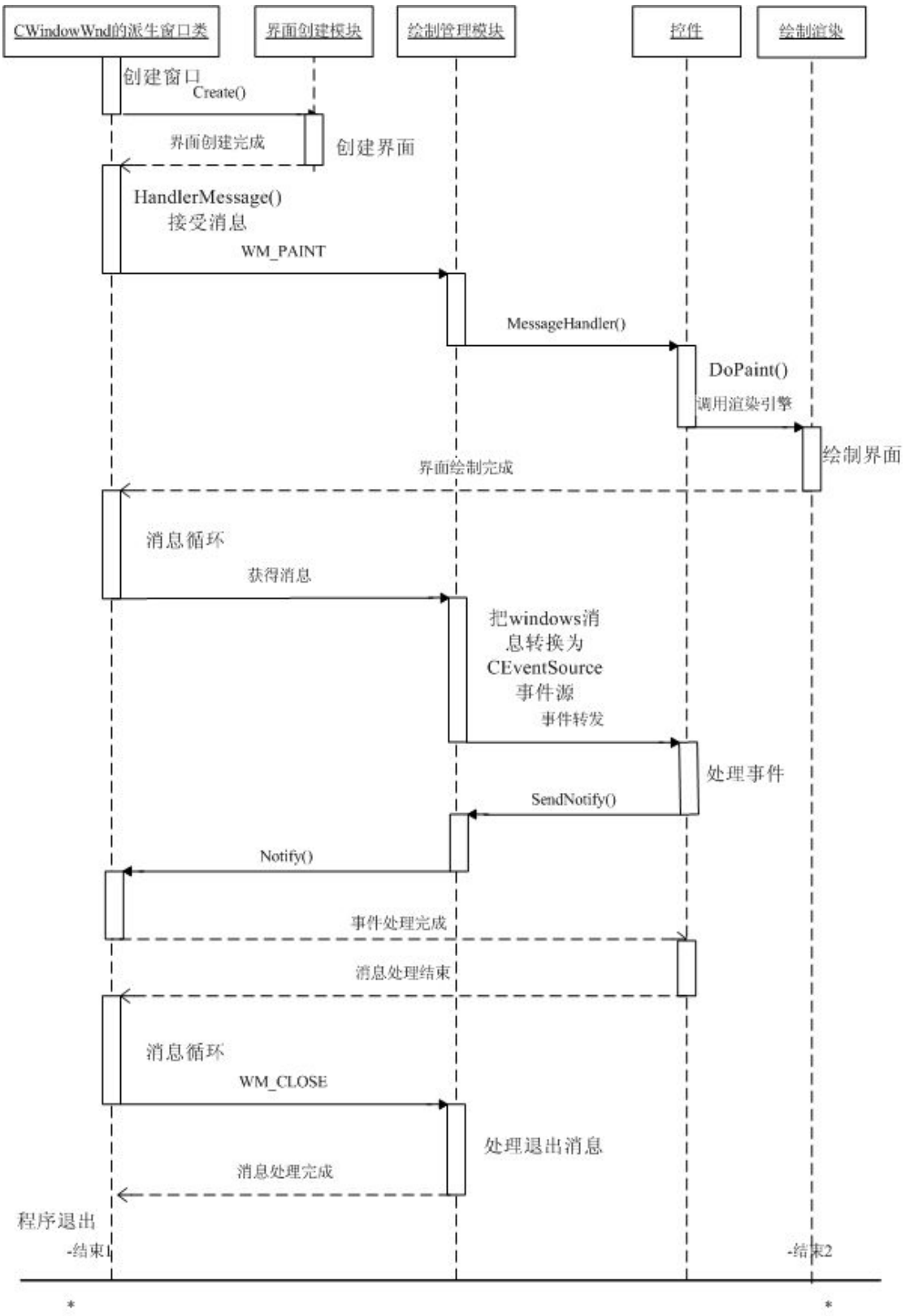
->CDialogBuilder::LoadFromFile (加载资源文件)

->CDialogBuilder::LoadFromFile (把资源文件写入内存)

->CDialogBuilder::Create (设置窗口和公共属性)

->CDialogBuilder::Parse (解析并生成控件)

上面的工作完成了 XML 数据的加载，并动态生成了控件，把控件加载到了控件列表，建立了与控件相关的 NameHash 中，接下来就是调用绘制管理模块绘制界面。最后程序进入消息循环。


CWindowWnd的派生窗口类
界面创建模块
绘制管理模块
控件
绘制渲染
创建窗口
Create()
界面创建完成
创建界面
HandlerMessage()
接受消息
WM_PAINT
MessageHandler()
DoPaint()
调用渲染引擎
绘制界面
界面绘制完成
消息循环
获得消息
把windows消
息转换为
CEventSource
事件源
事件转发
处理事件
SendNotify()
Notify()
事件处理完成
消息处理结束
消息循环
WM_CLOSE
处理退出消息
消息处理完成
程序退出
-结束1
-结束2
*
*

1.4 模块实现

1.4.1 基础结构模块

CPoint 结构：

CPoint 结构表示程序界面中的一个点，有四个构造函数。继承自 tagPOINT 结构体。有 2 个 LONG 类型的数据 x 和 y，分别是横坐标和纵坐标。

CSize 结构：

CSize 结构表示矩形的宽和高，有四个构造函数。继承自 tagSIZE 结构体，有 2 个 LONG 类型的数据 cx 和 cy，分别是宽和高。

CRect 结构：

CRect 结构表示程序界面中的一个矩形，继承自 tagRECT 结构体，有 4 个 LONG 类型的数据 left、top、right 和 bottom，表示了矩形左上和右下两个顶点的坐标值。

CStdString 结构：

CStdString 结构是在程序中使用最多的一种数据结构，用来表示字符串。该结构通过运算符重载和一些函数实现了很多传统字符串所不具有的功能，任何需要使用字符串的地方都用该结构体替代，使得程序对字符串的处理变得简单。

STRINGorID 结构：

为了对 dll 资源和压缩资源进行统一处理，通过重载构造函数的方法把资源 ID 和文件名称字符串抽象为 STRINGorID 结构。

CStdStringMap 结构：

CStdStringMap 结构用列表来存储字符串 CStdString 和数据 Data 的映射，TITEM 结构体是列表元素，包含字符串 CStdString 和 Data 数据。

CStdPtrArray 结构：

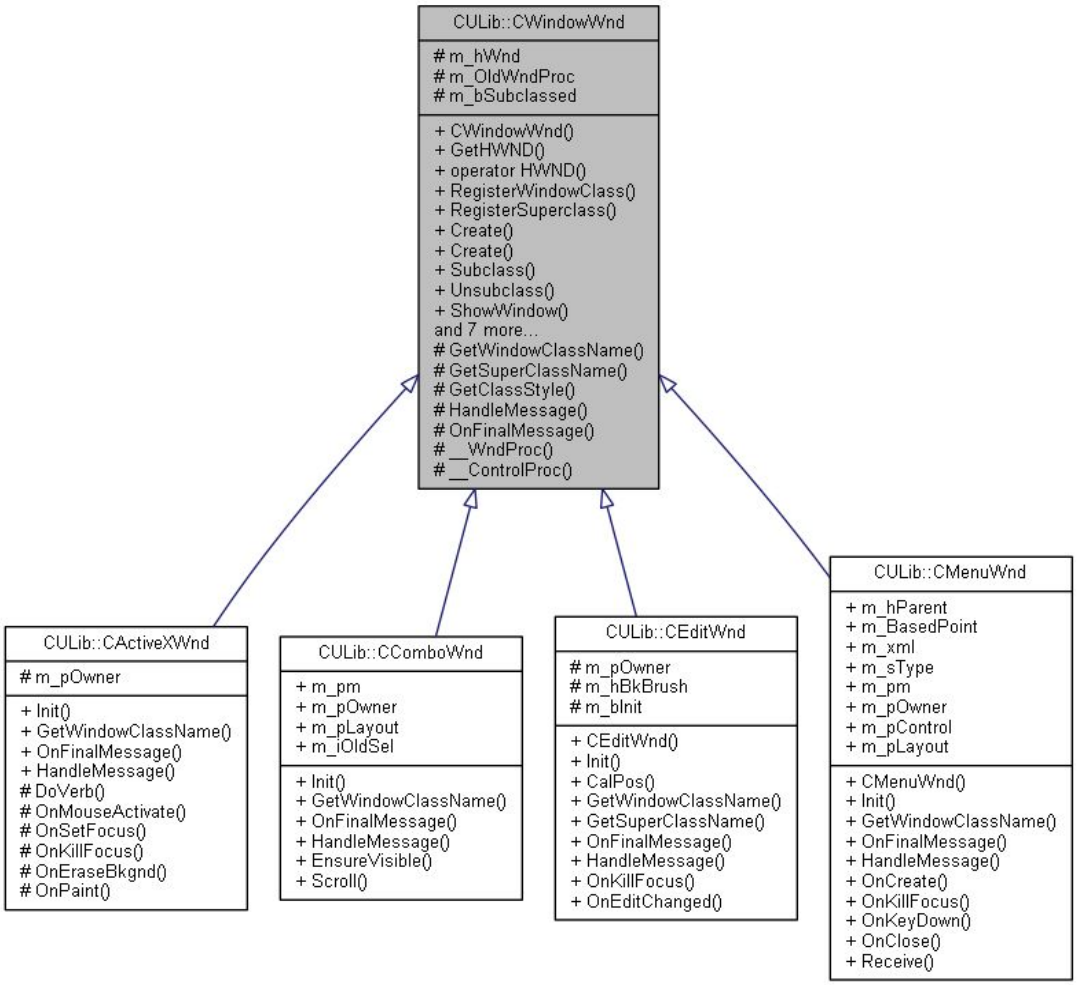
CStdPtrArray 结构用来表示一个指针数组，程序中不同类型的指针都可以放到 CStdPtrArray 结构中，方便了程序对指针的管理。

CStdValArray 结构：

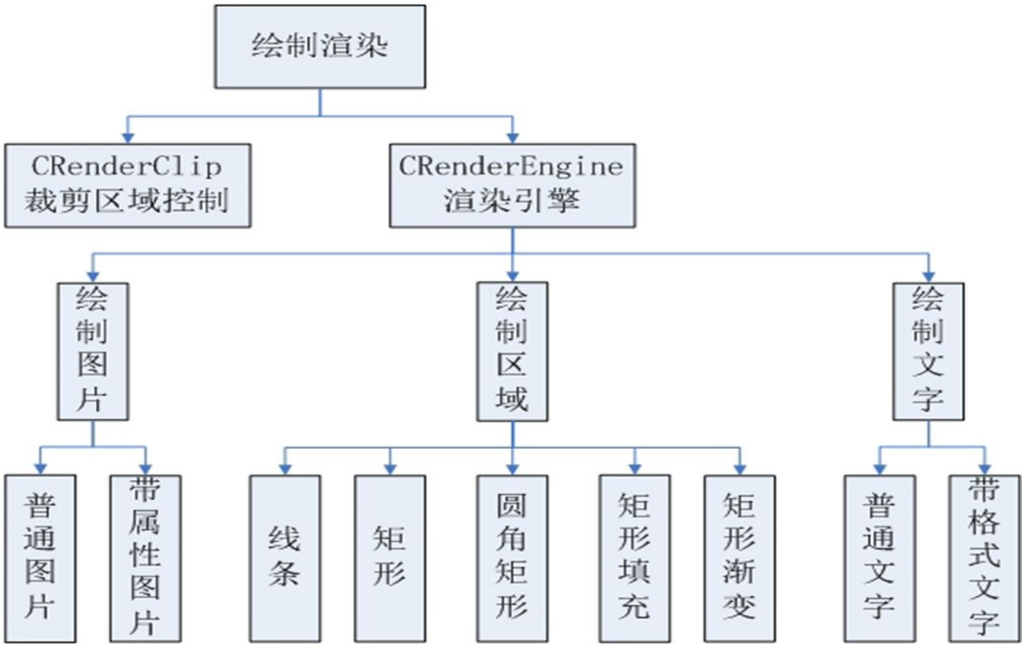
CStdValArray 结构用来表示一个值的数组，它的作用和上述指针数组的作用类似，也是为了对不同数据类型的数据进行统一管理。

CWindowWnd 结构：

CWindowWnd 是程序中所有窗口的基类继承关系，任何具有窗口的控件和主窗口都继承自该类。该结构实现了窗口的创建、窗口的超类化和子类化等功能。应用程序至少有一个主窗口，用户的消息处理函数在该窗口过程中被调用。CWindowWnd 类是用户使用该界面库时接触的第一个类，用户的主窗口会继承该类和通知接口 INotifyUI。CWindowWnd 类使得用户程序具有普通窗口的特征，通知接口使得用户程序可以监听界面中控件的事件，并对控件相应的事件进行处理。下图中的子类为三个不同的控件所用，有些控件在与用户的交互中会产生一个小的临时窗口，这时就可以借助 CWindowWnd 结构来实现。

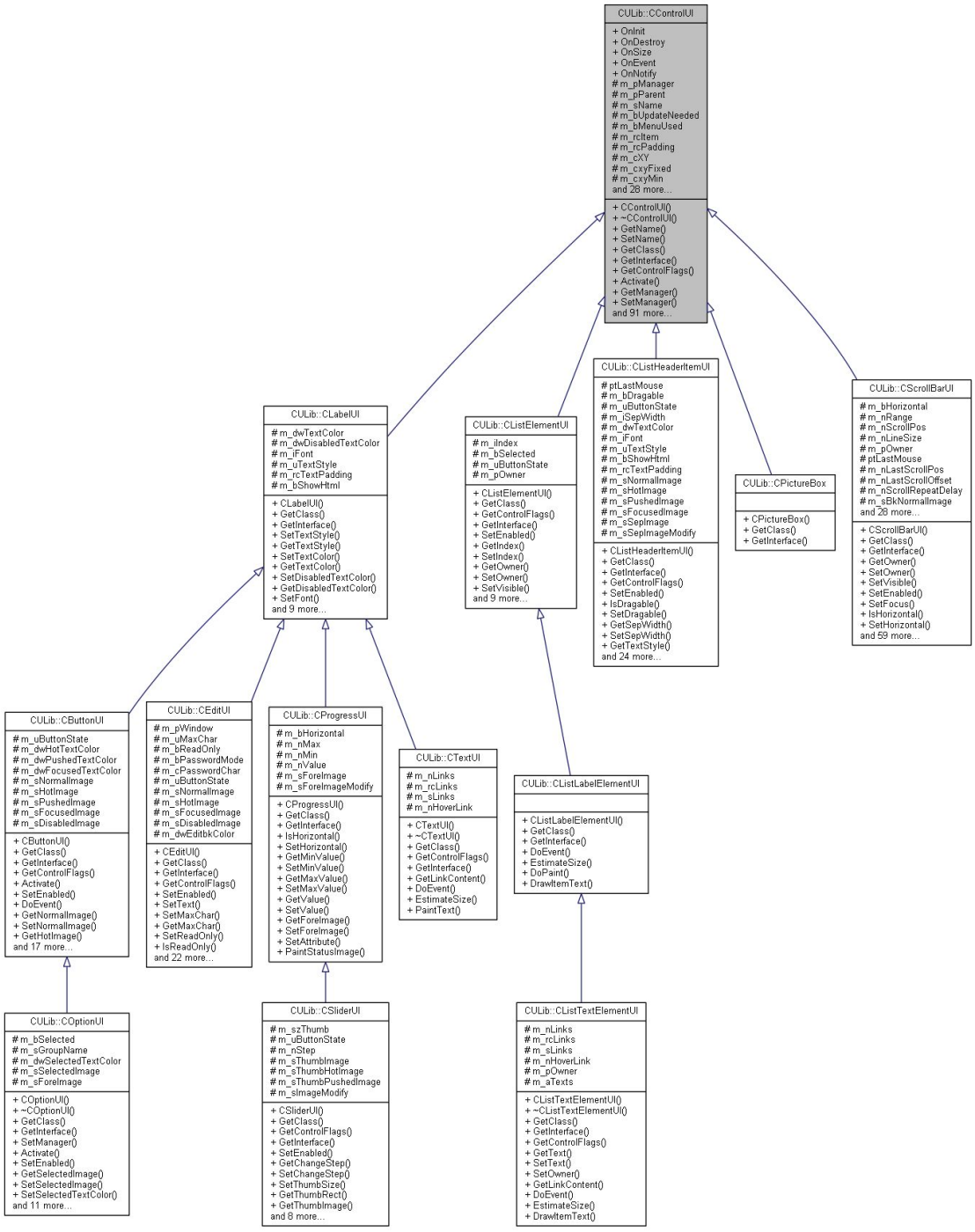


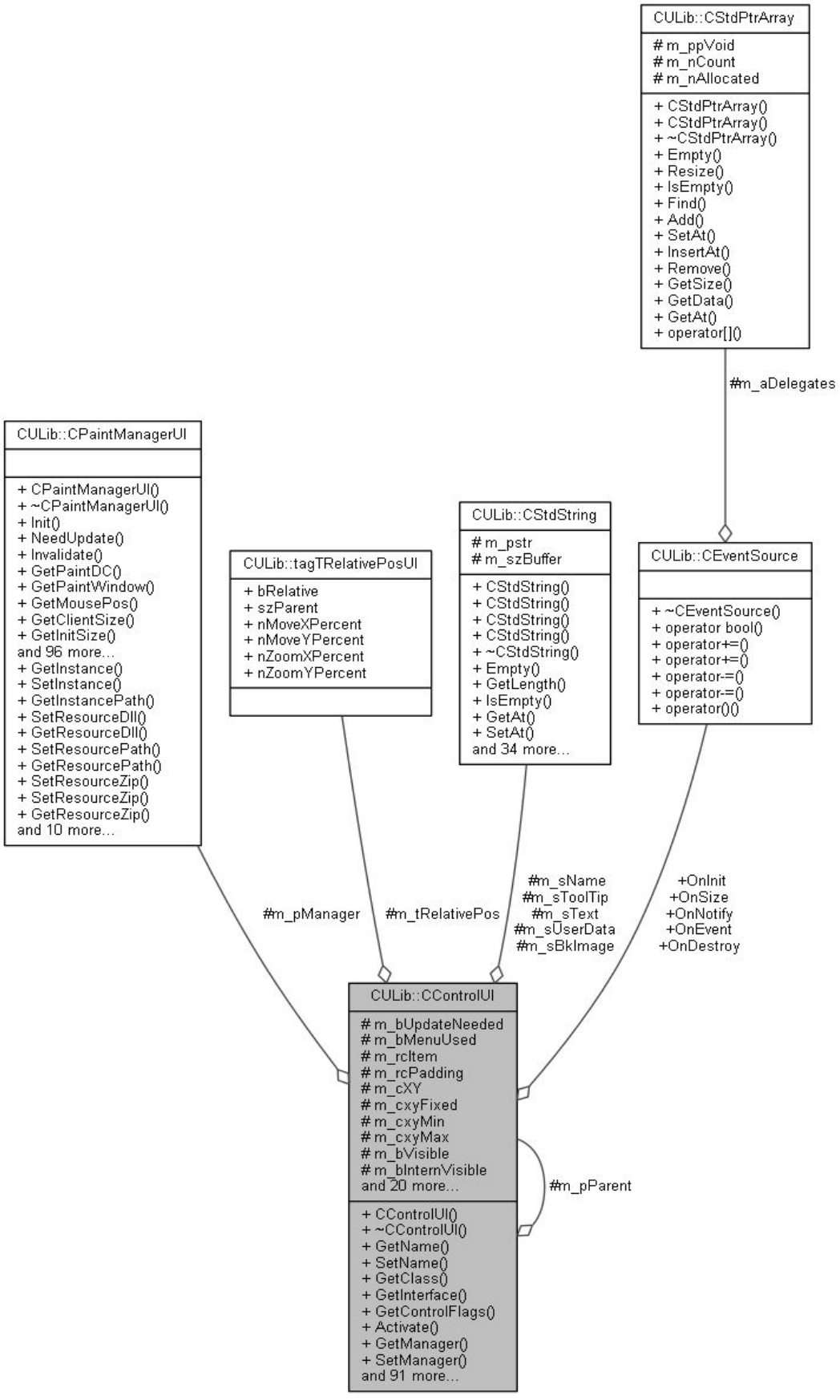
1.4.2 绘制渲染模块



绘制渲染模块有两个子模块，绘制区域剪裁 CRenderClip 和渲染引擎 CRenderEngine。CRenderC 的作用是，设置在 dc 上绘制的区域，防止子控件绘制到父控件外侧。它有四个函数，用来产生剪切区域。CRenderEngine 的作用是绘制各种资源，LoadImage 函数读取文件，资源和 zip 包中图像文件数据到内存，该函数调用 stb_image 库中的函数加载数据文件并转化后复制到 DIB 中。DrawImage 函数绘制图像，DrawImageString 按标识字符串绘制图像。GenerateBitmap 函数产生一个 BMP 位图并返回 BMP 句柄。DrwaHtmlText 绘制 "mini-html" 标识字符串，其余函数是对普通绘图函数的封装。

1.4.3 控件模块


DuiLib::CControlUI
+ OnInit
+ OnDestroy
+ OnSize
+ OnEvent
+ OnNotify
# m_pManager
# m_pParent
# m_sName
# m_bUpdateNeeded
# m_bMenuUsed
# m_rcItem
# m_rcPadding
# m_cXY
# m_cxyFixed
# m_cxyMin
and 28 more...
+ CControlUI()
+ ~CControlUI()
+ GetName()
+ SetName()
+ GetClass()
+ GetInterface()
+ GetControlFlags()
+ Activate()
+ GetManager()
+ SetManager()
and 91 more...
DuiLib::CLabelUI
m_dwTextColor
m_dwDisabledTextColor
m_iFont
m_uTextStyle
m_rcTextPadding
m_bShowHtml
+ CLabelUI()
+ GetClass()
+ GetInterface()
+ SetTextStyle()
+ GetTextStyle()
+ SetTextColor()
+ GetTextColor()
+ SetDisabledTextColor()
+ GetDisabledTextColor()
+ SetFont()
and 9 more...
DuiLib::CListElementUI
m_iIndex
m_bSelected
m_uButtonState
m_pOwner
+ CListElementUI()
+ GetClass()
+ GetControlFlags()
+ GetInterface()
+ SetEnabled()
+ GetIndex()
+ SetIndex()
+ GetOwner()
+ SetOwner()
+ SetVisible()
and 9 more...
DuiLib::CListHeaderItemUI
ptLastMouse
m_bDragable
m_uButtonState
m_iSepWidth
m_dwTextColor
m_iFont
m_uTextStyle
m_bShowHtml
m_rcTextPadding
m_sNormalImage
m_sHotImage
m_sPushedImage
m_sFocusedImage
m_sSepImage
m_sSepImageModify
+ CListHeaderItemUI()
+ GetClass()
+ GetInterface()
+ GetControlFlags()
+ SetEnabled()
+ IsDragable()
+ SetDragable()
+ GetSepWidth()
+ SetSepWidth()
+ GetTextStyle()
and 24 more...
DuiLib::CPictureBox
+ CPictureBox()
+ GetClass()
+ GetInterface()
DuiLib::CScrollBarUI
m_bHorizontal
m_nRange
m_nScrollPos
m_nLineSize
m_pOwner
ptLastMouse
m_nLastScrollPos
m_nLastScrollOffset
m_nScrollRepeatDelay
m_sBkNormalImage
and 28 more...
+ CScrollBarUI()
+ GetClass()
+ GetInterface()
+ GetOwner()
+ SetOwner()
+ SetVisible()
+ SetEnabled()
+ SetFocus()
+ IsHorizontal()
+ SetHorizontal()
and 59 more...
DuiLib::CButtonUI
m_uButtonState
m_dwHotTextColor
m_dwPushedTextColor
m_dwFocusedTextColor
m_sNormalImage
m_sHotImage
m_sPushedImage
m_sFocusedImage
m_sDisabledImage
+ CButtonUI()
+ GetClass()
+ GetInterface()
+ GetControlFlags()
+ Activate()
+ SetEnabled()
+ DoEvent()
+ GetNormalImage()
+ SetNormalImage()
+ GetHotImage()
and 17 more...
DuiLib::CEditUI
m_pWindow
m_uMaxChar
m_bReadOnly
m_bPasswordMode
m_cPasswordChar
m_uButtonState
m_sNormalImage
m_sHotImage
m_sFocusedImage
m_sDisabledImage
m_dwEditbkColor
+ CEditUI()
+ GetClass()
+ GetInterface()
+ GetControlFlags()
+ SetEnabled()
+ SetText()
+ SetMaxChar()
+ GetMaxChar()
+ SetReadOnly()
+ IsReadOnly()
and 22 more...
DuiLib::CProgressUI
m_bHorizontal
m_nMax
m_nMin
m_nValue
m_sForeImage
m_sForeImageModify
+ CProgressUI()
+ GetClass()
+ GetInterface()
+ IsHorizontal()
+ SetHorizontal()
+ GetMinValue()
+ SetMinValue()
+ GetMaxValue()
+ SetMaxValue()
+ GetValue()
+ SetValue()
+ GetForeImage()
+ SetForeImage()
+ SetAttribute()
+ PaintStatusImage()
DuiLib::CTextUI
m_nLinks
m_rcLinks
m_sLinks
m_nHoverLink
+ CTextUI()
+ ~CTextUI()
+ GetClass()
+ GetControlFlags()
+ GetInterface()
+ GetLinkContent()
+ DoEvent()
+ EstimateSize()
+ PaintText()
DuiLib::CListLabelElementUI
+ CListLabelElementUI()
+ GetClass()
+ GetInterface()
+ DoEvent()
+ EstimateSize()
+ DoPaint()
+ DrawItemText()
DuiLib::COptionUI
m_bSelected
m_sGroupName
m_dwSelectedTextColor
m_sSelectedImage
m_sForeImage
+ COptionUI()
+ ~COptionUI()
+ GetClass()
+ GetInterface()
+ SetManager()
+ Activate()
+ SetEnabled()
+ GetSelectedImage()
+ SetSelectedImage()
+ SetSelectedTextColor()
and 11 more...
DuiLib::CSliderUI
m_szThumb
m_uButtonState
m_nStep
m_sThumbImage
m_sThumbHotImage
m_sThumbPushedImage
m_sImageModify
+ CSliderUI()
+ GetClass()
+ GetControlFlags()
+ GetInterface()
+ SetEnabled()
+ GetChangeStep()
+ SetChangeStep()
+ SetThumbSize()
+ GetThumbRect()
+ GetThumbImage()
and 8 more...
DuiLib::CListTextElementUI
m_nLinks
m_rcLinks
m_sLinks
m_nHoverLink
m_pOwner
m_aTexts
+ CListTextElementUI()
+ ~CListTextElementUI()
+ GetClass()
+ GetInterface()
+ GetControlFlags()
+ GetText()
+ SetText()
+ SetOwner()
+ GetLinkContent()
+ DoEvent()
+ EstimateSize()
+ DrawItemText()


CULib::CStdPtrArray
m_ppVoid
m_nCount
m_nAllocated
+ CStdPtrArray()
+ CStdPtrArray()
+ ~CStdPtrArray()
+ Empty()
+ Resize()
+ IsEmpty()
+ Find()
+ Add()
+ SetAt()
+ InsertAt()
+ Remove()
+ GetSize()
+ GetData()
+ GetAt()
+ operator[]()
#m_aDelegates
CULib::CPaintManagerUI
+ CPaintManagerUI()
+ ~CPaintManagerUI()
+ Init()
+ NeedUpdate()
+ Invalidate()
+ GetPaintDC()
+ GetPaintWindow()
+ GetMousePos()
+ GetClientSize()
+ GetInitSize()
and 96 more...
+ GetInstance()
+ SetInstance()
+ GetInstancePath()
+ SetResourceDll()
+ GetResourceDll()
+ SetResourcePath()
+ GetResourcePath()
+ SetResourceZip()
+ SetResourceZip()
+ GetResourceZip()
and 10 more...
CULib::tagTRelativePosUI
+ bRelative
+ szParent
+ nMoveXPercent
+ nMoveYPercent
+ nZoomXPercent
+ nZoomYPercent
CULib::CStdString
m_pstr
m_szBuffer
+ CStdString()
+ CStdString()
+ CStdString()
+ CStdString()
+ ~CStdString()
+ Empty()
+ GetLength()
+ IsEmpty()
+ GetAt()
+ SetAt()
and 34 more...
CULib::CEventSource
+ ~CEventSource()
+ operator bool()
+ operator+=()
+ operator+=()
+ operator-=()
+ operator-=()
+ operator()()
#m_pManager
#m_tRelativePos
#m_sName
#m_sToolTip
#m_sText
#m_sUserData
#m_sBkImage
+OnInit
+OnSize
+OnNotify
+OnEvent
+OnDestroy
CULib::CControlUI
m_bUpdateNeeded
m_bMenuUsed
m_rcItem
m_rcPadding
m_cXY
m_cxyFixed
m_cxyMin
m_cxyMax
m_bVisible
m_bInternVisible
and 20 more...
+ CControlUI()
+ ~CControlUI()
+ GetName()
+ SetName()
+ GetClass()
+ GetInterface()
+ GetControlFlags()
+ Activate()
+ GetManager()
+ SetManager()
and 91 more...
#m_pParent

1. 控件基类

控件基类 CControlUI 在整个控件体系中非常重要，如图 3.6 所示，它是所有控件的基类，也是组成控件树的基本元素，控件树中的所有节点都是一个 CControlUI。它基本包含了所有控件公共的属性，如：位置、大小、颜色、是否有焦点、是否被启用、是否隐藏等等、当然这个类中还提供了非常多的基础函数，用于重载来实现子控件，如获取控件名称 GetClassName、DoEvent 等等。为了方便从 xml 中直接解析出控件的各个属性，这个类中还提供了一个 SetAttribute 方法，传入字符串的属性值名称和值对特定的属性进行设置，内部其实就是挨个比较字符串去完成的。另外每个控件中还有几个事件管理的对象,CEventSource，这些对象会在特定的时机被触发，如 OnNotify，并调用其中保存的各个回调函数。所有控件都要在 DoEvent 函数中对消息进行处理，当某个消息没有被处理，会传到该控件的父类控件中进行处理，如果所有控件都没有处理此消息则交给 CControlUI 基类控件做默认处理。与控件相关的消息都以 UIEVENT_开头，基类控件的消息处理如下表所示。

表 3.1 CControlUI 控件消息处理

消息类型	消息处理
UIEVENT_SETCURSOR	调用 SetCursor 捕获鼠标
UIEVENT_SETFOCUS	设置焦点，重绘
UIEVENT_KILLFOCUS	失去焦点，重绘
UIEVENT_MOUSENTER	鼠标进入，设置定时器
UIEVENT_MOUSLEAVE	鼠标离开，取消定时器
UIEVENT_TIMER	定时器超时，显示 ToolTip 提示，发送”timer”通知
UIEVENT_CONTEXTMENU	显示系统菜单，发送”menu”通知

从实现的角度来看，控件可以分为两种类型。一种是没有产生新窗口的控件，包括：标签控件，按钮控件，选项控件，滑块控件，滚动条控件，进度条控件和文本控件。另一种是需要产生新窗口的控件，包括：编辑框控件，下拉框控件，列表控件，菜单控件和组件控件。控件的继承关系如图 3.5 所示。

第一种控件的特点是控件的实现简单与用户的交互少并且控件的所有元素已经全部在界面中一次性表现出来了。大部分都是用来静态显示文本或图片信息，如标签、按钮、选项文本等控件。而滑块、滚动条、进度条等虽然实现复杂，但是在交互的过程中不需要产生新的元素。所有这些简单的控件都可以组合起来以实现复杂的控件，如 option 控件与绘制管理中的选项组列表相结合实现的选项卡功能。

第二种控件的特点是实现比较复杂，如编辑框控件和组件控件，与用户交互过程中必须产生新的元素，如下拉框、列表框、菜单等控件。这些控件通过封装一些具有窗口的控件以供用户使用，这样做的目的是统一系统对所有控件的处理。

2. 控件消息处理

从整个系统的角度来看一共可以分为三种消息：Windows 消息、界面库消息、用户层消

息。Windows 消息是 Windows 系统向程序发送的消息，界面库获取该种消息并把它转化为界面库对应的消息 tagTEventUI。界面库中的不同控件根据 tagTeventUI 消息的类型执行不同的动作，当需要与用户程序进行交互时，就向用户程序发送用户层消息。用户就是通过用户层消息与界面库进行交互。因此要学会使用控件必须清楚消息在控件内部的流转机制，以下就是控件对界面库消息的处理和与用户层消息的映射。

表 3.2　CLabel 控件消息处理

消息类型	消息处理
UIEVENT_SETFOCUS	设置焦点
UIEVENT_KILLFOCUS	失去焦点

表 3.3　CButton 控件消息处理

消息类型	消息处理
UIEVENT_KEYDOWN	发送”click”通知
UIEVENT_BUTTONDOWN	更新状态
UIEVENT_MOUSEMOVE	更新状态
UIEVENT_BUTTONUP	发送”click”通知
UIEVENT_CONTEXTMENU	发送”menu”通知
UIEVENT_MOUSENTER	更新状态，调用 CLabel::DoEvent()
UIEVENT_MOUSLEAVE	更新状态，调用 CLabel::DoEvent()
UIEVENT_SETCURSOR	调用 SetCursor 捕获鼠标

表 3.4　COptionUI 控件消息处理

消息类型	消息处理
UIEVENT_KEYDOWN	更新状态，发送”selectChanged”通知
UIEVENT_BUTTONUP	更新状态，发送”selectChanged”通知

注 COptionUI 控件调用 CButtonUI 的 DoEvent 函数

表 3.5　CProgressUI 控件消息处理

消息类型	消息处理
UIEVENT_CONTEXTMENU	返回

注 CProgressUI 控件调用 CButtonUI 的 DoEvent 函数

表 3.6　CEditUI 控件消息处理

消息类型	消息处理
UIEVENT_SETCURSOR	调用 SetCursor 捕获鼠标
UIEVENT_WINDOWSIZE	调用 CPaintManagerUI::SetFocusNedded()，发送”killfocus”通知
UIEVENT_SCROLLWHEEL	如果有编辑框窗口则返回
UIEVENT_SETFOCUS	产生 CEditWnd 编辑框窗口
UIEVENT_KILLFOCUS	更新状态并调用 CLabel::DoEvent()
UIEVENT_BUTTONDOWN	激活 CEditWnd 编辑框窗口
UIEVENT_MOUSEENTER	更新状态，调用 CLabel::DoEvent()
UIEVENT_MOUSELEAVE	更新状态，调用 CLabel::DoEvent()

表 3.7　CSliderUI 控件消息处理

消息类型	消息处理
UIEVENT_BUTTONDOWN	更新状态
UIEVENT_BUTTONUP	发送”valuechanged”通知
UIEVENT_SCROLLWHELL	发送”valuechanged”通知
UIEVENT_MOUSEMOVE	计算当前 value 值
UIEVENT_SERCURSOR	调用 SetCursor 捕获鼠标
UIEVENT_MOUSEENTER	更新状态，调用 CControlUI::DoEvent()
UIEVENT_MOUSELEAVE	更新状态，调用 CControlUI::DoEvent()
UIEVENT_CONTEXTMENU	返回

表 3.8　CTextUI 控件消息处理

消息类型	消息处理
UIEVENT_SETCURSOR	调用 SetCursor 捕获鼠标
UIEVENT_BUTTONDOWN	更新状态
UIEVENT_BUTTONUP	发送”link”通知
UIEVENT_CONTEXTMENU	返回
UIEVENT_MOUSEMOVE	更新状态
UIEVENT_MOUSELEAVE	更新状态，调用 CLabel::DoEvent()

表 3.9　CListUI 控件消息处理

消息类型	消息处理
UIEVENT_SETFOCUS	更新状态
UIEVENT_KILLFOCUS	更新状态
UIEVENT_KEYDOWN	查找选中项，发送”itemselect”通知
UIEVENT_SCROLLWHELL	查找选中项，发送”itemselect”通知

注 默认调用 CVerticalLayoutUI::DoEvent()

注 CListBodyUI 控件调用 CListUI 控件的 DoEvent()

表 3.10　CListHeaderItemUI 控件消息处理

消息类型	消息处理
UIEVENT_SETFOCUS	更新状态，调用基类 DoEvent()
UIEVENT_KILLFOCUS	更新状态，调用基类 DoEvent()
UIEVENT_BUTTONDOWN	更新状态，发送”headerclick”通知
UIEVENT_BUTTONUP	更新状态
UIEVENT_MOUSEMOVE	设置需要更新
UIEVENT_SETCURSOR	调用 SetCursor 捕获鼠标
UIEVENT_MOUSEENTER	更新状态，调用 CControlUI::DoEvent()
UIEVENT_MOUSELEAVE	更新状态，调用 CControlUI::DoEvent()

表 3.11　CListElementUI 控件消息处理

消息类型	消息处理
UIEVENT_DBLCLICK	发送”itemactive”通知
UIEVENT_KETDOWN	若是 RETUEN 键则发送”itemactive”通知

注 默认调用拥有者的 DoEvent()

表 3.12　CListLabelElementUI 控件

消息类型	消息处理
UIEVENT_BUTTONDOWN	选中 item，发送”itemclick”通知
UIEVENT_MOUSEMOVE	返回
UIEVENT_BUTTONUP	返回
UIEVENT_MOUSEENTER	更新状态，调用 CListElementUI::DoEvent()
UIEVENT_MOUSELEAVE	更新状态，调用 CListElementUI::DoEvent()

表 3.13　CListTextElementUI 控件消息处理

消息类型	消息处理
UIEVENT_SETCURSOR	调用 SetCursor 捕获鼠标
UIEVENT_BUTTONUP	发送”link”通知
UIEVENT_MOUSEMOVE	设置选中项，更新状态
UIEVENT_MOUSELEAVE	更新状态，调用 CListElementUI::DoEvent()

注 默认调用 CListTextElementUI::DoEvent()

表 3.14　CListContainerElementUI 控件消息处理

消息类型	消息处理
UIEVENT_DBLCLICK	发送”itemactive”通知
UIEVENT_KETDOWN	若是 RETUEN 键则发送”itemactive”通知
UIEVENT_BUTTONDOWN	发送”itemclick”通知
UIEVENT_BUTTONUP	返回
UIEVENT_MOUSEMOVE	返回
UIEVENT_MOUSEENTER	更新状态，调用拥有自己控件的 DoEvent()
UIEVENT_MOUSELEAVE	更新状态，调用拥有自己控件的 DoEvent()

表 3.15　CMenuElementUI 控件消息处理

消息类型	消息处理
UIEVENT_MOUSEENTER	产生 CMenuWnd 窗口或选中 item，发送”itemselect”通知
UIEVENT_BUTTONDOWN	产生 CMenuWnd 窗口或发送”itemactivate”通知

注 CMenuUI 控件调用其父类 CListUI 控件的 DoEvent()

注 默认调用 CListContainerElementUI::DoEvent()

表 3.16　CScrollBarUI 控件消息处理

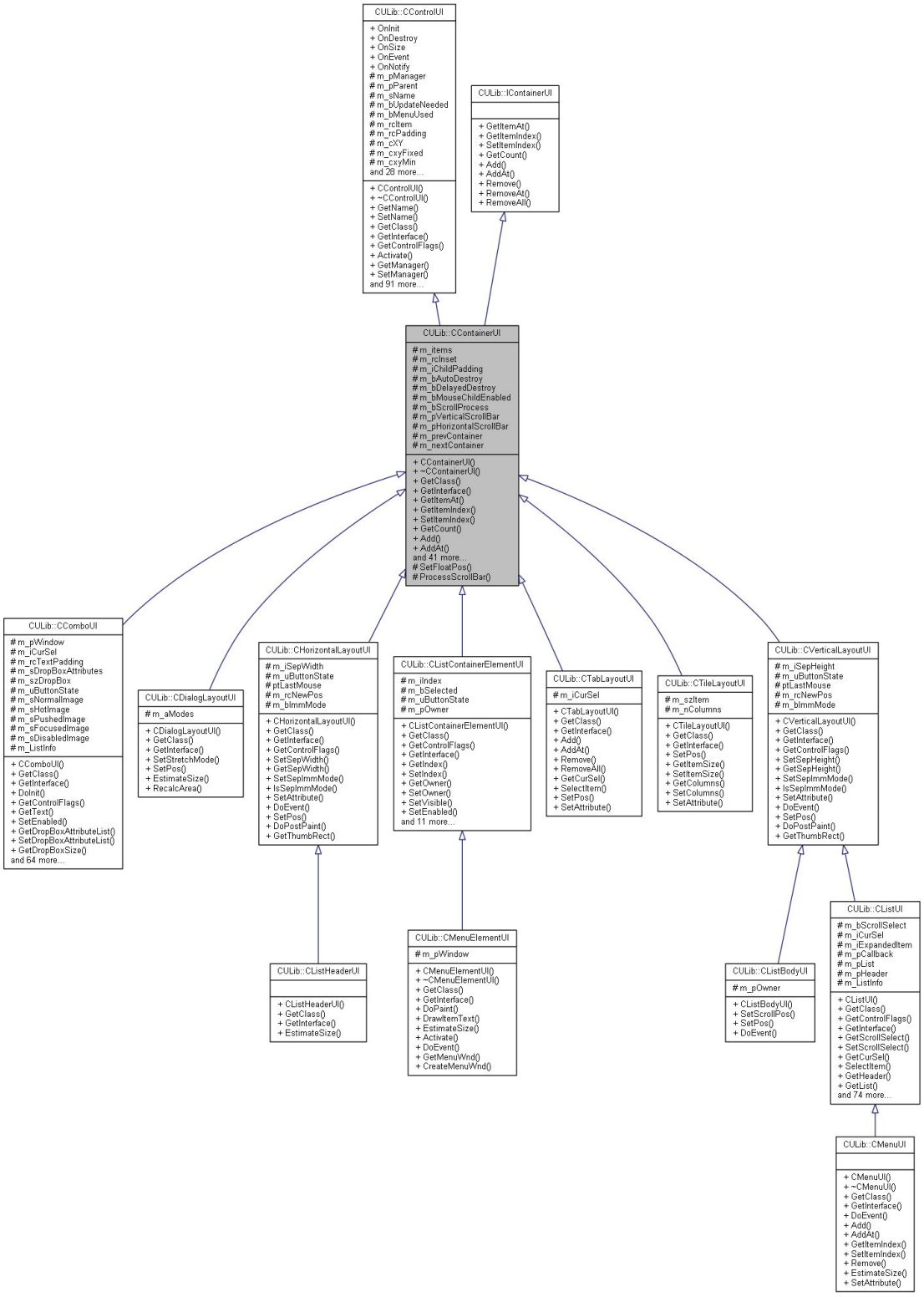
消息类型	消息处理
UIEVENT_SETFOCUS	返回
UIEVENT_KILLFOCUS	返回
UIEVENT_BUTTONDOWN	设置定时器计算并更新滚动条当前值，发送”scroll”通知
UIEVENT_BUTTONUP	取消定时器，更新状态
UIEVENT_MOUSEMOVE	计算并更新滚动条当前值，更新状态
UIEVENT_CONTEXTMENU	返回
UIEVENT_MOUSEENTER	更新状态
UIEVENT_MOUSELEAVE	更新状态
UIEVENT_TIMER	更新滚动条当前值，发送”scroll”通知

注 默认调用其拥有者的 DoEvent()

表 3.17　CComboUI 控件消息处理

消息类型	消息处理
UIEVENT_SETFOCUS	更新状态，调用基类 DoEvent()
UIEVENT_KILLFOCUS	更新状态，调用基类 DoEvent()
UIEVENT_BUTTONDOWN	更新状态
UIEVENT_BUTTONUP	更新状态
UIEVENT_KEYDOWN	查找选中项，发送”itemselect”通知
UIEVENT_SCROLLWHELL	查找选中项，发送”itemselect”通知
UIEVENT_CONTEXTMENU	返回
UIEVENT_MOUSEENTER	更新状态
UIEVENT_MOUSELEAVE	更新状态

1.4.4 布局管理模块



布局管理基类：

有了基本的控件之后，我们就需要布局管理器将它们管理起来，CContainerUI 是所有布局管理器的基类，其内部用一个数组来保存所有的 CControlUI 对象，CContainerUI 也继承自 CControlUI。在 CContainer 容器里面主要实现了以下几个功能：

子控件查找：

CContainerUI::FindControl

子控件的生命周期管理：

在 Remove 的时候销毁或者交给 CPaintManagerUI 延迟销毁

滚动条：

所有布局管理器都支持滚动条，在其内部会对键盘和鼠标滚轮事件进行处理，对其内部所有的控件元素调整位置，最后在绘制的时候实现滚动的效果。

绘制：

由于布局管理器中有很多元素，所以为了加快容器的绘制，绘制的时候会获取器真正需要绘制的区域，如果子控件不在此区域中，那么就不绘制了。

布局管理器也要处理系统消息，消息类型与控件一样都是以 UIEVENT_开头。并在 DoEvent 函数中进行消息的处理，如果消息没有被处理会被传入到 CContainerUI 的 DoEvent 中进行处理，在 DoEvent 中会调用 CControlUI 的 DoEvent，CContainerUI 与布局管理器的消息处理如下所示所示。

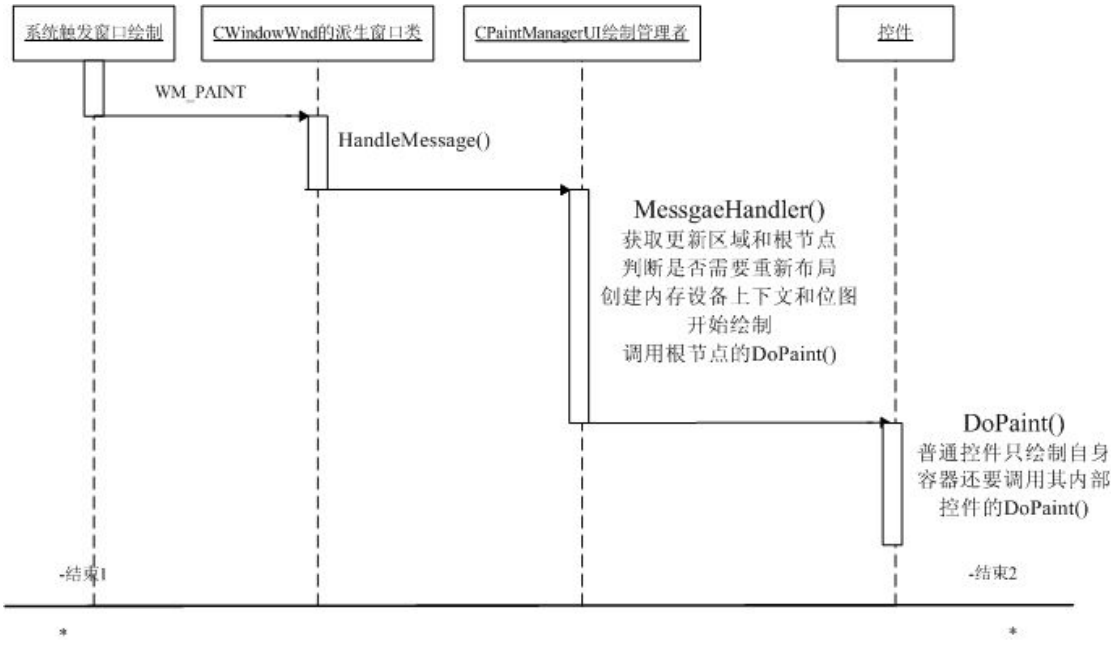
表 3.18　CContainer 容器消息处理

消息类型	消息处理
UIEVENT_SETFOCUS	设置焦点，重绘
UIEVENT_KILLFOCUS	失去焦点，重绘
UIEVENT_KEYDOWN	根据按键向不同方向操纵滚动条

表 3.19　LayoutUI 消息处理

消息类型	消息处理
UIEVENT_BUTTONDOWN	把自己加入绘制回调列表中
UIEVENT_BUTTONUP	把自己从绘制回调列表中移除并设置需要更新
UIEVENT_MOUSEMOVE	刷新区域
UIEVENT_SETCURSOR	调用 SetCursor 捕获鼠标

1.4.5 绘制管理模块



当所有基本的控件和布局管理器都准备好之后，我们就只要将它们管理起来，这样一个基本的控件库就完成了，而这个管理就是 CPaintManagerUI 来负责的。在 CULib 中，一个 Windows 的原生窗口和一个 CPaintManagerUI 一一对应。其主要负责以下几个内容：控件管理，消息管理，转化并分发 Windows 原生的窗口消息。为了实现这几个功能，用到了以下一些数据结构：

m_pRoot：保存根控件的节点

m_mNameHash：保存控件名称和控件对象指针的对应关系

m_mOptionGroup：保存控件相关的 Group，用于实现 Option 控件

TFontInfo：用来管理字体资源

m_ImageHash：用来管理图片资源

m_DefaultAttrHash：管理控件的默认属性

除了以上提到的结构外还有句柄管理结构：如与绘制相关的句柄和资源相关的句柄；与消息处理相关的结构：如计时器列表，预处理消息列表；记录界面状态的结构：如是否需要更新，是否捕获鼠标等等。

CPaintManagerUI 还有一些比较重要的函数就是根据不同条件查找控件，包括通过点的坐标查找控件，通过名字查找控件，通过对象查找控件等等。

CPaintManagerUI 对控件的管理都集中在 MessageHandler 函数中，其内部会对很多窗口消息进行处理，并将其分发到对应的控件上去。包括对 WM_PAINT、WM_CLOSE、WM_MOUSEMOVE、WM_LBUTTONDOWN 等的处理。另外 CPaintManagerUI 还提供了其他几种用于处理消息的方法：

Notifier：

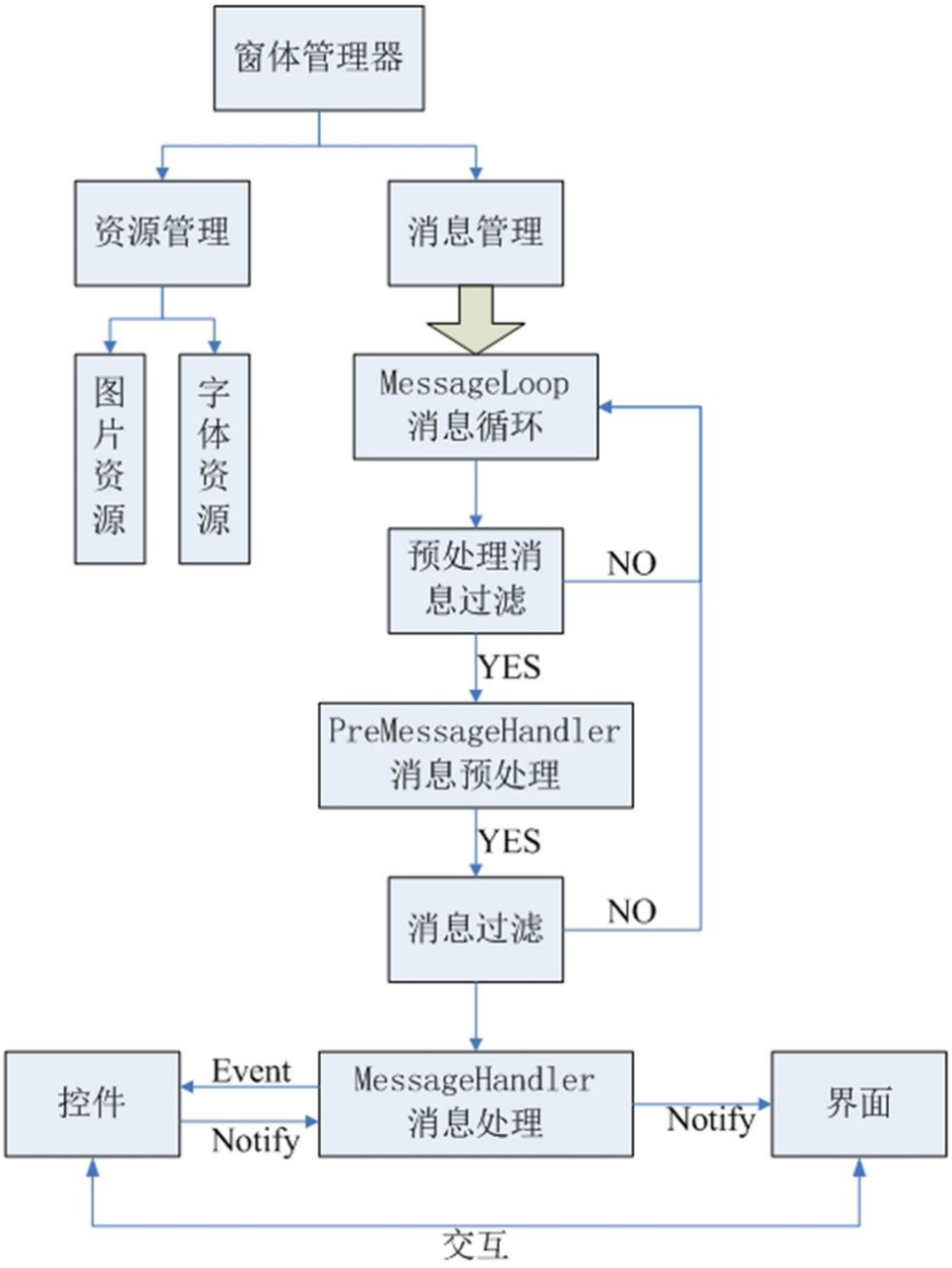
用户的主窗口类可以继承 INotifyUI 接口，并实现 OnNotify 方法。CPaintManagerUI 在对消息进行处理时通过 Notify 函数向用户发送通知，用户程序就可以在 OnNotify 中对消息进行处理。

PreMessageFilter：

消息预处理，把处理消息的类添加到消息预处理列表中，每次接收到消息时都会调用每个消息预处理列表中的类方法。

PostPaint：

绘制完成后的回调函数。

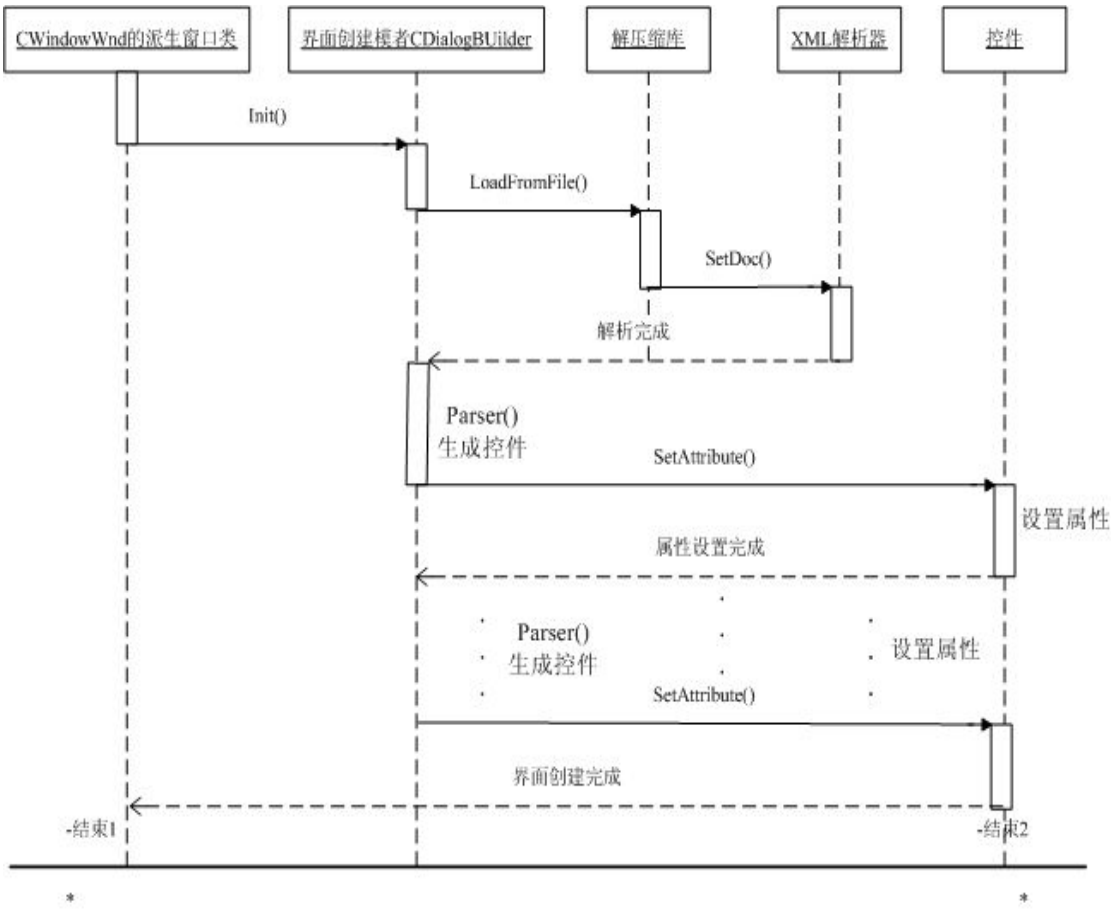


1.4.6 界面创建模块

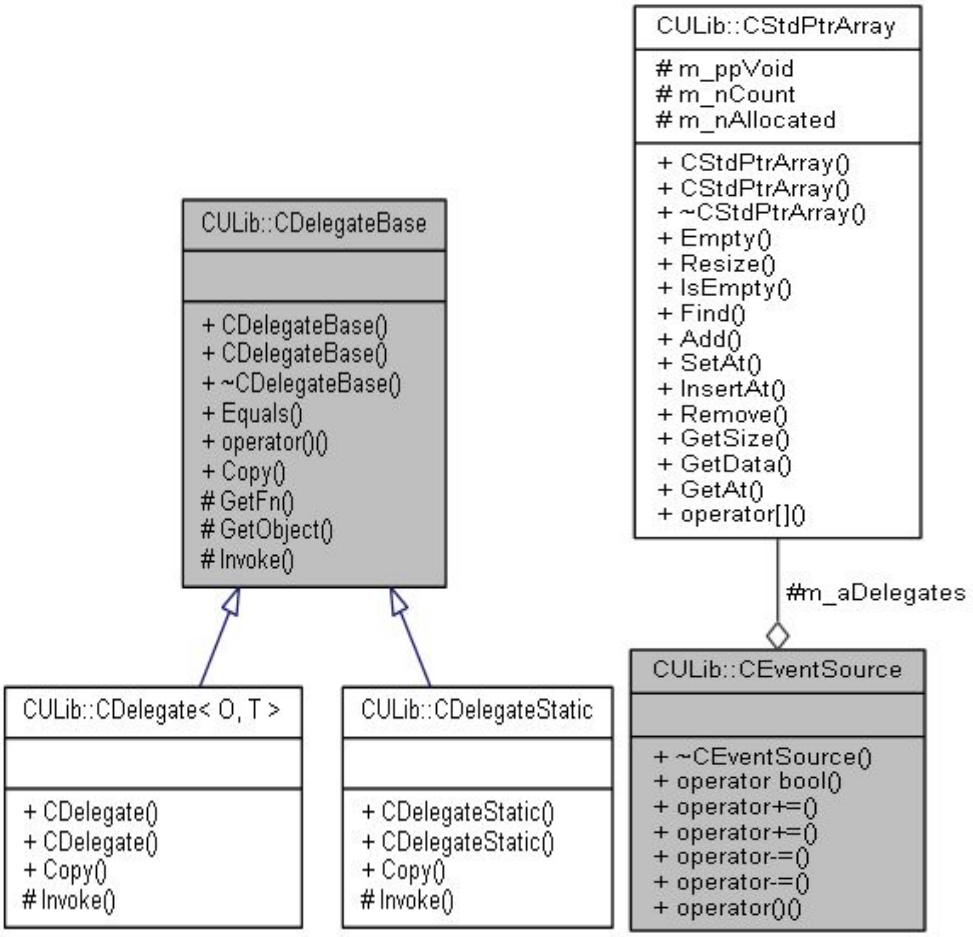
现在已经有了控件管理和控件库，接下来就要考虑如何通过配置文件来生成界面。CULib 中的界面是由 xml 描述文件和资源图片两部分生成的，xml 文件描述窗口中控件的布局和样式，资源图片就是界面最终的呈现。

我们把这些资源文件放在一个文件夹中，这样就形成了基础的皮肤包。当然我们还可以将其压缩成一个 zip 包，从而加快 IO 访问或者将其制作为一个二进制的 dll 资源供程序加载。资源中最关键的就是 xml 描述文件了，一个 xml 描述文件对应着一个窗口的信息，如：控件的类型和样式等等。

为了通过配置文件自动创建界面，CULib 提供了一个类：CDialogBuilder。这个类提供了从皮肤包中的 xml 中创建界面的方法：CDialogBuilder::Create。在其内部调用了 CMarkup 解析器来解析 xml 文件，并依此创建各个控件。除了创建控件，还将一些可以复用的资源提取出来放入 CPaintManagerUI 中统一管理，如字体和图片等等。界面创建的过程如下图所示。


CWindowWnd的派生窗口类
界面创建模者CDialogBUilder
解压缩库
XML解析器
控件
Init()
LoadFromFile()
SetDoc()
解析完成
Parser()
生成控件
SetAttribute()
设置属性
属性设置完成
Parser()
生成控件
设置属性
SetAttribute()
界面创建完成
-结束1
-结束2
*
*

1.4.7 事件委托模块



CDelegateBase 是委托模式中的虚基类，它的两个子类 CDelegate 和 CDelegateStatic 是具体的委托。CDelegateStatic 是静态委托，委托者为空，但有执行过程。

两个产生委托的函数是：

CDelegate MakeDelegate(O* pObject, bool (T::* pFn)(TEventUI&))

CDelegateStatic MakeDelegate(bool (*pFn)(TEventUI&))

CEventSource 是事件源，是对委托的封装，每个控件都有 5 种事件源，分别为 OnInit、OnDestroy、OnSize、OnEvent 和 OnNotify。在 CULib 中和控件相关的消息都被封装在一个类型为 tagTEventUI 结构体的 event 变量中。每个控件都会调用 OnEvent 事件源中的委托方法来处理 event 事件。并且在函数 DoEvent()中针对不同的事件调用 OnNotify 委托方法以及向用户程序发送通知。因此应用程序可以通过向以上 5 种事件源添加委托的方法来与界面库进行交互。

加入一个委托者

operator+= (CDelegateBase& d) ； operator+= (FnType pFn)

删除一个委托者

operator-= (CDelegateBase& d) ； operator-= (FnType pFn)

响应事件进行处理

operator() (TEventUI& event)

1.5 控件使用

用户在 xml 文件中描述好控件的风格和布局后控件的绘制就交给界面库去完成了，用户只需要关心与控件的交互。用户与控件的交互也就是用户响应控件的各种事件，有两种方法可以实现：

第一，实现用户窗口实例中的 Notify 函数，控件会向该函数发送用户层消息，各种控件的用户层消息如表 5.1 所示。在该函数中对不同控件的不同事件消息进行处理。

第二，使用委托技术，首先使用 FindControl 函数找到添加委托的控件：CControlUI* CPaintManagerUI::FindControl(LPCTSTR pstrName)。之后调用 MakeDelegate 函数生成委托：CDelegate<O, T> MakeDelegate(O* pObject, bool (T::* pFn)(void*))，其中的 pFn 就是控件事件的处理函数。最后把委托添加到控件的事件源对象 OnNotify 上：void CEventSource::operator+= (CDelegateBase& d)。

表 5.1 用户层消息

消息	说明	发送控件
click	鼠标点击	CButtonUI
dropdown	下拉显示	CComboUI
headerclick	点击列标题	CListHeaderItemUI
itemactivate		CListElementUI、CListContainerElementUI
itemclick	单击选项	CListLabelElementUI、CListContainerElementUI
itemselect	选择选项	CComboUI、CListUI
killfocus	失去焦点	CControlUI
link		CTextUI、CListTextElementUI
menu		CButtonUI、CControlUI
return	回车	CEditWnd、CRichEditUI
scroll	滚动	CScrollBarUI
selectchanged	变更选项	COptionUI
setfocus	获得焦点	CControlUI
showactivex		CActiveXUI
textchanged	文本被改变	CEditWnd
tabselect	标签页被选中	CTabLayoutUI
timer		CControlUI
valuechanged	值发生变化	CSliderUI
windowinit	窗体初始化	

1.6 界面配置

1. 属性配置

表 5.2 window 属性表

名称	类型	说明
size	SIZE	窗口初始化大小
caption	RECT	标题栏大小
roundcorner	SIZE	窗口圆角大小
mininfo	SIZE	窗口最小大小
maxinfo	SIZE	窗口最大大小
disabledfontcolor	DWORD	默认的 disabled 字体颜色
defaultfontcolor	DWORD	默认的字体颜色
linkfontcolor	DWORD	默认的 link 字体颜色
linkhoverfontcolo	DWORD	默认的 linkhoverfont 字体颜色
selectedcolor	DWORD	默认的 selected 字体颜色

表 5.3 Control 属性表

名称	类型	说明
name	STRING	控件名字，同一窗口内必须唯
pos	RECT	位置
padding	RECT	外边距
bkcolor	DWORD	背景颜色
bkcolor2	DWORD	背景渐变色 2
bkcolor3	DWORD	背景渐变色 3
bordercolor	DWORD	边框颜色
focusbordercolor	DWORD	获得焦点时边框的颜色
borderround	SIZE	边框圆角半径
bordersize	INT	边框大小
bkimage	STRING	背景图片
width	INT	宽度
height	INT	高度
minwidth	INT	最小宽度
minheight	INT	最小高度
maxwidth	INT	最大宽度
maxheight	INT	最大高度
text	STRING	显示文本
tooltip	STRING	鼠标悬浮提示

userdata	STRING	自定义标识
enabled	BOOL	是否响应用户操作
mouse	BOOL	是否响应鼠标操作
float	BOOL	是否使用绝对定位
shortcut	CHAR	对应的快捷键
menu	BOOL	是否需要菜单
colorhsl	BOOL	控件的颜色是否随窗口的 hsl 变化而变化

表 5.4　Container 属性表

名称	类型	说明
inset	RECT	容器的内边距
vscrollba	BOOL	是否使用竖向滚动条
hscrollbar	BOOL	是否使用横向滚动条
childpadding	INT	子控件之间的额外距离

注 该属性表包含表 5.3　Control 属性表中的属性

表 5.5　VerticalLayout 属性表

名称	类型	说明
sepheight	INT	分隔符高度,正负表示分隔符在顶部还是底部
sepimm	BOOL	拖动分隔符是否立即改变大小

注 该属性表包含表 5.4　Container 属性表中的属性

表 5.6　HorizontalLayout、ListHeader 属性表

名称	类型	说明
sepwidth	INT	分隔符宽,正负表示分隔符在左边还是右边
sepimm	BOOL	拖动分隔符是否立即改变大小

注 该属性表包含表 5.4　Container 属性表中的属性

表 5.7　TileLayout 属性表

名称	类型	说明
columns	INT	列数
itemsize	SIZE	子项固定大

注 该属性表包含表 5.4　Container 属性表中的属性

表 5.8　TabLayout 属性表

名称	类型	说明
selectedid	INT	默认选中的页面 id

注 该属性表包含表 5.4　Container 属性表中的属性

表 5.9　ActiveX 属性表

名称	类型	说明
clsid	STRING	activex 的 clsid
modulename	STRING	activex 从指定位置加载
delaycreate	BOOL	是否需要延迟创建 activex

注 该属性表包含表 5.3　Control 属性表中的属性

表 5.10　Label、Text 属性表

名称	类型	说明
align	STRING	文字对齐方式,取值 left、right、center、top、bottom
endellipsis	BOOL	句末显示不全是否使用...代替
font	INT	字体 id
textcolor	DWORD	字体颜色，0 表示使用默认字体颜色
disabledtextcolor	DWORD	disabled 字体颜色，0 表示使用默认 disabled 字体颜色
textpadding	RECT	文字显示的边距
showhtml	BOOL	是否使用类 html 富文本绘制

注 该属性表包含表 5.3　Control 属性表中的属性

表 5.11　Combo 属性表

名称	类型	说明
textpadding	RECT	文字显示的边距
normalimage	STRING	普通状态图片
hotimage	STRING	鼠标悬浮的状态图片
pushedimage	STRING	鼠标按下的状态图片
focusedimage	STRING	获得焦点时的状态图片
disabledimage	STRING	禁用的状态图片
dropboxsize	STRING	弹出框大小设置
itemfont	INT	item 的字体 id
itemalign	STRING	item 对齐方式,取值 left、right、center
itemendellipsis	BOOL	item 句末显示不全是否使用...代替
itemtextpadding	RECT	item 文字显示的边距
itemtextcolor	DWORD	item 字体颜色
itembkcolor	DWORD	item 背景颜色
itembkimage	STRING	item 背景图片
itemaltbk	BOOL	item 是否使用隔行交替背景
itemselectedtextcolor	DWORD	item 被选中时的字体颜色
itemselectedbkcolo	DWORD	item 被选中时的背景颜色
itemselectedimage	STRING	item 被选中时的背景图片
itemhottextcolor	DWORD	item 鼠标悬浮时的字体颜色
itemhotbkcolor	DWORD	item 鼠标悬浮时的背景颜色
itemhotimage	STRING	item 鼠标悬浮时的背景图片
itemdisabledtextcolor	DWORD	item 禁用时的字体颜色

itemdisabledbkcolor	DWORD	item 禁用时的背景颜色
itemdisabledimage	STRING	item 禁用时的背景图片
itemlinecolor	DWORD	item 行分割线颜色
itemshowhtml	BOOL	item 是否使用类 html 富文本绘制
multiexpanding	BOOL	是否支持多个 item 同时打开

注 该属性表包含表 5.4 Container 属性表中的属性

表 5.12 Button 属性表

名称	类型	说明
normalimage	STRING	普通状态图片
hotimage	STRING	鼠标悬浮的状态图片
pushedimage	STRING	鼠标按下的状态图片
focusedimage	STRING	获得焦点时的状态图片
disabledimage	STRING	禁用的状态图片
hottextcolor	DWORD	鼠标悬浮字体颜色，0 表示不使用此颜色
pushedtextcolor	DWORD	鼠标按下字体颜色，0 表示不使用此颜色
focusedtextcolor	DWORD	获得焦点字体颜色，0 表示不使用此颜色

注 该属性表包含表 5.10 Label、Text 属性表中的属性

表 5.13 Option 属性表

名称	类型	说明
selectedtextcolor	DWORD	选中状态字体颜色，0 表示不使用此颜色
group	STRING	所属组的名称，可不设
selected	BOOL	是否选中

注 该属性表包含表 5.12 Button 属性表中的属性

表 5.14 Progress 属性表

名称	类型	说明
foreimage	STRING	前景图片
hor	BOOL	水平或垂直
min	INT	进度最小值
max	INT	进度最大值
value	INT	进度值
isstretchfore	BOOL	指定前景图片是否缩放显示

注 该属性表包含表 5.10 Label、Text 属性表中的属性

表 5.15 Slider 属性表

名称	类型	说明
foreimage	STRING	前景图片
hor	BOOL	水平或垂直
min	INT	进度最小值
max	INT	进度最大值

value	INT	进度值
thumbimage	STRING	拖动滑块普通状态图片
thumbhotimage	STRING	拖动滑块鼠标悬浮状态图片
thumbpushedimage	STRING	拖动滑块鼠标按下状态图片
thumbsize	SIZE	拖动滑块大小
step	INT	进度步长

注 该属性表包含表 5.10 Label、Text 属性表中的属性

表 5.16 Edit 属性表

名称	类型	说明
readonly	BOOL	是否只读
password	BOOL	是否显示密码字符
maxchar	INT	输入字符最大长度
normalimage	STRING	普通状态图片
hotimage	STRING	鼠标悬浮的状态图片
pushedimage	STRING	鼠标按下的状态图片
focusedimage	STRING	获得焦点时的状态图片
disabledimage	STRING	禁用的状态图片
nativebkcolor	DWORD	windows 原生 edit 控件的背景颜色

注 该属性表包含表 5.10 Label、Text 属性表中的属性

表 5.17 ScrollBar 属性表

名称	类型	说明
button1normalimage	STRING	左或上按钮普通状态图片
button1hotimage	STRING	左或上按钮鼠标悬浮状态图片
button1pushedimage	STRING	左或上按钮鼠标按下状态图片
button1disabledimage	STRING	左或上按钮禁用状态图片
button2normalimage	STRING	右或下按钮普通状态图片
button2hotimage	STRING	右或下按钮鼠标悬浮状态图片
button2pushedimage	STRING	右或下按钮鼠标按下状态图片
button2disabledimage	STRING	右或下按钮禁用状态图片
thumbnormalimage	STRING	滑块普通状态图片
thumbhotimage	STRING	滑块鼠标悬浮状态图片
thumbpushedimage	STRING	滑块鼠标按下状态图片
thumbdisabledimage	STRING	滑块禁用状态图片
railnormalimage	STRING	滑块中间标识普通状态图片
railhotimage	STRING	滑块中间标识鼠标悬浮状态图片
railpushedimage	STRING	滑块中间标识鼠标按下状态图片
raildisabledimage	STRING	滑块中间标识禁用状态图片
bknormalimage	STRING	背景普通状态图片
bkhotimage	STRING	背景鼠标悬浮状态图片
bkpushedimage	STRING	背景鼠标按下状态图片
bkdisabledimage	STRING	背景禁用状态图片

hor	BOOL	水平或垂直
linesize	INT	滚动一行的大小
range	INT	滚动范围
value	INT	滚动位置
showbutton1	BOOL	是否显示左或上按钮
showbutton2	BOOL	是否显示右或下按钮

注 该属性表包含表 5.3 Control 属性表中的属性

表 5.18 ListLabelElement、ListTextElement 属性表

名称	类型	说明
selected	BOOL	是否选中

注 该属性表包含表 5.3 Control 属性表中的属性

表 5.19 ListContainerElement 属性表

名称	类型	说明
selected	BOOL	是否选中

注 该属性表包含表 5.4 Container 属性表中的属性

表 5.20 List 属性表

名称	类型	说明
header	BOOL	是否显示表头
headerbkimage	STRING	表头背景图片
scrollselect	BOOL	是否随滚动改变选中项
itemfont	INT	item 的字体 id
itemalign	STRING	item 对齐方式,取值 left、right、center
itemendellipsis	BOOL	item 句末显示不全是否使用...代替
itemtextpadding	RECT	item 文字显示的边距
itemtextcolor	DWORD	item 字体颜色
itembkcolor	DWORD	item 背景颜色
itembkimage	STRING	item 背景图片
itemaltbk	BOOL	item 是否使用隔行交替背景
itemselectedtextcolor	DWORD	item 被选中时的字体颜色
itemselectedbkcolo	DWORD	item 被选中时的背景颜色
itemselectedimage	STRING	item 被选中时的背景图片
itemhottextcolor	DWORD	item 鼠标悬浮时的字体颜色
itemhotbkcolor	DWORD	item 鼠标悬浮时的背景颜色
itemhotimage	STRING	item 鼠标悬浮时的背景图片
itemdisabledtextcolor	DWORD	item 禁用时的字体颜色
itemdisabledbkcolor	DWORD	item 禁用时的背景颜色
itemdisabledimage	STRING	item 禁用时的背景图片
itemlinecolor	DWORD	item 行分割线颜色

itemshowhtml	BOOL	item 是否使用类 html 富文本绘制
multiexpanding	BOOL	是否支持多个 item 同时打开

注 该属性表包含表 5.5 VerticalLayout 属性表中的属性

表 5.21 ListHeaderItem 属性表

名称	类型	说明
dragable	BOOL	是否可拖动改变大小
sepwidth	INT	分隔符宽
align	STRING	文字对齐方式,取值 left、right、cente
endellipsis	BOOL	句末显示不全是否使用...代替
font	INT	字体 id
textcolor	DWORD	字体颜色，0 表示使用默认字体颜色
textpadding	RECT	文字显示的边距
showhtml	BOOL	是否使用类 html 富文本绘制
normalimage	STRING	普通状态图片
hotimage	STRING	鼠标悬浮的状态图片
pushedimage	STRING	鼠标按下的状态图片
focusedimage	STRING	获得焦点时的状态图片
sepimage	STRING	拖动条图片

注 该属性表包含表 5.3 Control 属性表中的属性

表 5.22 RichEdit 属性表

名称	类型	说明
nativebkcolor	DWORS	输入时编辑框背景色

注 该属性表包含表 5.4 Container 属性表中的属性

2. 布局管理

水平布局

用 HorizontalLayout 容器，子控件设定好 width，则子控件会按照从左到右的顺序排列。

垂直布局

用 VerticalLayout 容器，子控件设定好 height，则子控件会按照从上到下的顺序排列。

间隔

在两个控件之间放置一个 Control 元素，要保持水平间隔就设置 width，要保持垂直间距就设置 height。

左右等分

在 HorizontalLayout 容器中放置多个控件，不设置 width，则它们会自动平分宽度。

上下等分

在 VerticalLayout 容器中放置多个控件，不设置 width，则它们会自动平分高度。

锚定在右边

在 HorizontalLayout 容器中先放置一个 Control 但不设置 width，再放置要锚定的控件，

设定好 width。

锚定在下方

在 VerticalLayout 容器中先放置一个 Control 但不设置 height，再放置要锚定的控件，设定好 height。

左右不动，中间拉伸

在 HorizontalLayout 容器中放置 3 个控件，左右控件固定 width，中间的控件不设置 width。

上下不动，中间拉伸

在 VerticalLayout 容器中放置 3 个控件，上下控件固定 height，中间的控件不设置 height。

九宫格布局

在 VerticalLayout 容器中放置三个 VerticalLayout 容器，上下固定 height；然后在每个 VerticalLayout 容器里再放置三个 HorizontalLayout 容器，左右固定 width 即可。当然也可以