Dulib消息处理过程

1. 先是在注册、创建窗口，在函数CWindowWnd::Create中完成。里面分别调用了windows API，RegisterClass、CreateWindowEx，此时系统会向WndProc直接发送WM\_NCCREATE消息，在窗口过程函数CWindowWnd::\_\_WndProc中，此时保存了窗口指针pThis，也即（来自于CreateWindowEx函数的最后一个参数( 也就是CWindowWnd对象指针了)）。后面，程序将收到WM\_CREATE消息，这时，因为CWindowWnd::\_\_WndProc中PThis指针不为NULL，因此将跳转到派生窗口类中进行处理。

附一点关于WM\_CREATE消息的内容(WM\_NCCREATE是在WM\_CREATE之前发送的一条消息)：

CreateWindow()是在运行后、返回前，发送消息WM\_CREATE，并阻塞！

WM\_CREATE消息直接发给消息处理函数WndProc()执行！而不是通过消息队列（GetMessage、DispatchMessage等）

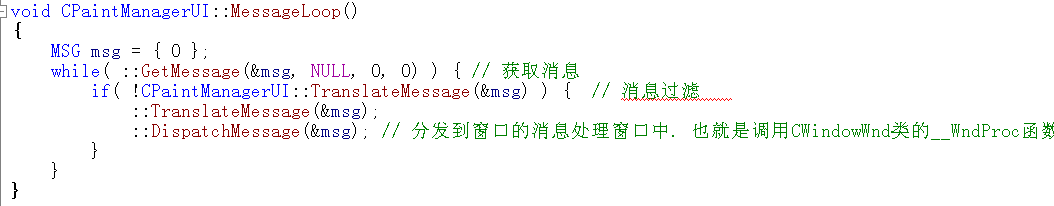
其实就是CreateWindow()直接拿消息WM\_CREATE去调用WndProc()

如果WndProc()返回0，CreateWindow()正常返回hWnd。 如果WndProc()返回-1，CreateWindow()销毁窗口，并返回NULL。

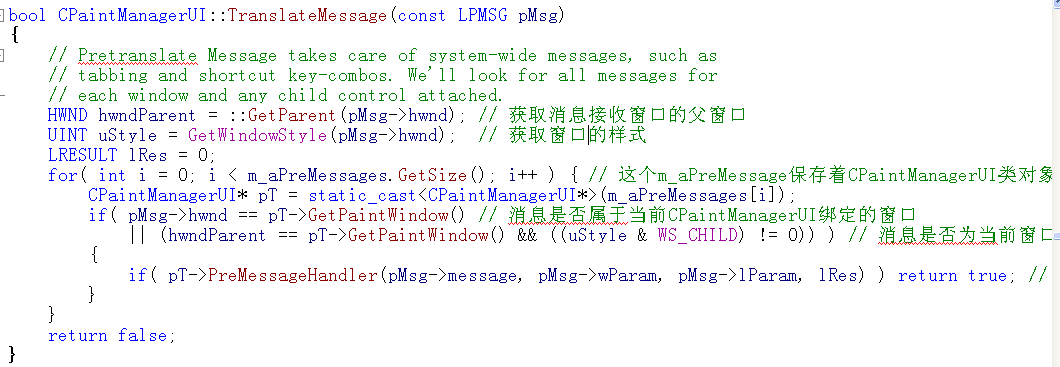
所以WM\_CREATE时，CreateWindow()内部已经创建好窗口了，但还没有返回，还没有

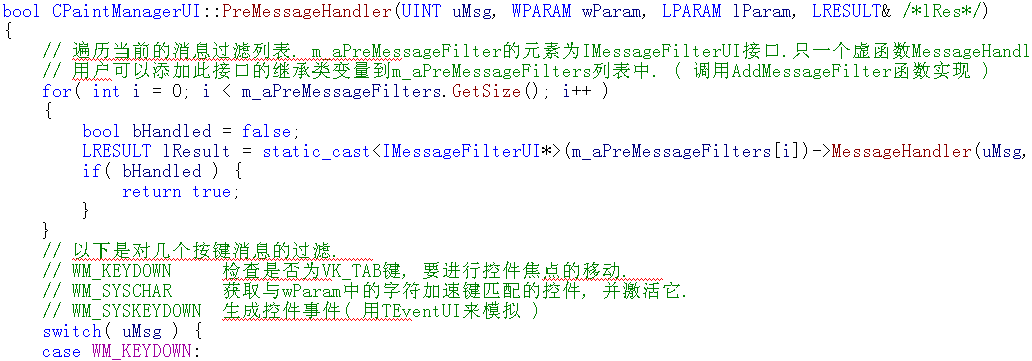
hWnd！即窗口真正还没有创建结束。这时这里不能有对窗口上的控件操作的函数，不然就会“程序执行了非法操作”。比如GetDlgItem等

1. 接下来进入消息循环CPaintManagerUI::MessageLoop()了

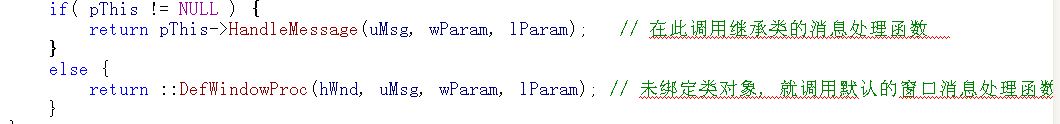


1. 在CPaintManagerUI::TranslateMessage(&msg)进行了消息预处理和调用CPaintManagerUI::PreMessageHandler进行过滤。





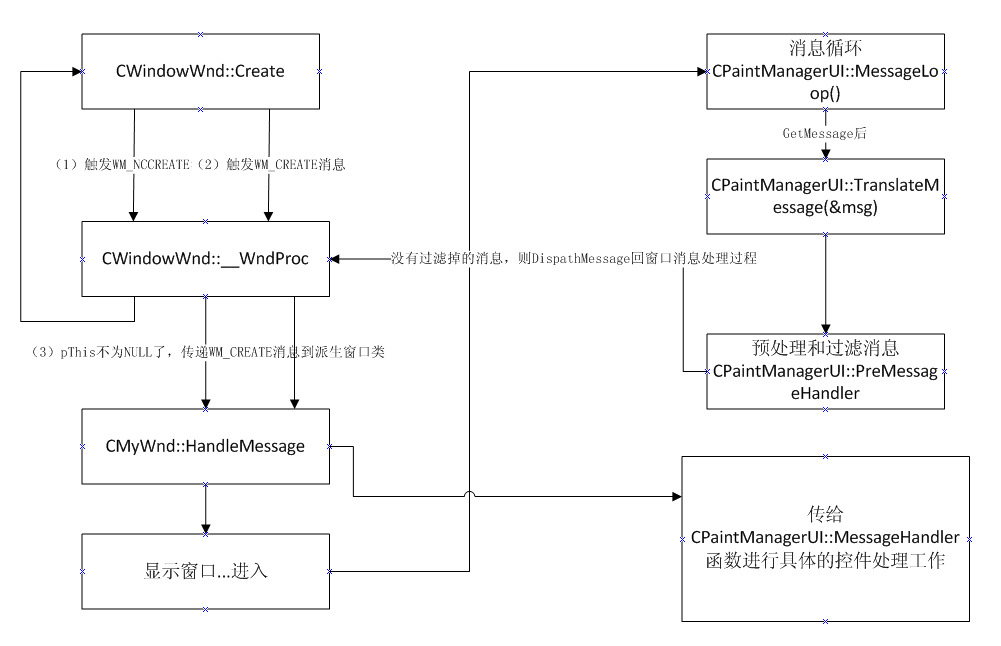
1. 没有过滤和预处理的消息则由DispathMessage，发送到CWindowWnd::\_\_WndProc，在该函数中，调用了窗口派生类的HandleMessage



1. 这是示例app的窗口派生类的HandleMessage函数：



其消息流程图如下所示：



参考文档：

<http://blog.csdn.net/yimsungnam/article/details/6935855>