送给初学者的礼物：C 游戏编程起源连载三 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/133/2021_2022__E9_80_81_E7_BB_99_E5_88_9D_E5_c97_133521.htm 第三章跟踪你的窗口和使用GDI简介如果你看过了头两篇连载，这次我们将学习WINDOWS GDI（图形设备接口）和其它一些相关的东西，象响应用户输入和处理Windows产生的一些消息。至于显示图形，我们将接触三个课题：文本显示，绘制象素，显示位图。我们先来研究一下几个Windows消息的细节。重复的话：你需要C语言的基础知识，最好看过上两章。由于本章将使你能做一个具体的图形DEMO，有一个源代码例程附在本章后面。是用Visual C＋＋写的和编译的。在连载一里，我们创建和注册了一个窗口类，其中有一行定义了窗口的风格（功能），是这个样子：sampleClass.style = CS_DBLCLKS | CS_OWNDC | CS_HREDRAW | CS_VREDRAW.// standard settings其中三个属性是很一般的，但这个CS_OWNDC，需要解释一下。设备上下文是一个结构，是一个表现一组图形对象和属性的结构，还有一些输出设备的设置和属性。使用设备上下文允许你直接操纵图形，不用考虑低级细节。Windows GDI是一个图形翻译系统，是介于应用程序和图形硬件之间的一层。GDI可以输出到任意的兼容设备，不过最常使用的设备是视频监视器、图形硬拷贝设备（如打印机或绘图仪），或者是内存中的图元文本。GDI函数能够绘制直线、曲线、封闭的图形和文本。所有访问GDI的Windows函数都需要一个设备上下文句柄作为参数。这是非常容易做到的。你若想得到一个窗口的设备上下文句柄，你可以用这个

函数： HDC GetDC(HWND hWnd // handle to a window).很简单，所有你做的是，把要操作的窗口的句柄传递给它，然后返回一个设备上下文句柄。如果你传递的是NULL，将返回整个屏幕的设备上下文（DC，以后都用DC表示）句柄。如果函数调用失败，将返回NULL。 处理显示图形的DC类型，称作显示DC，处理打印的，称作打印DC；处理位图数据的，称作内存DC，还有其它一些设备DC。感觉有点复杂吧，不要紧，这是Windows，它的主要功能就是迷惑群众。一旦我们接触一些代码，就不会觉得难了。 当你结束使用DC时，一定要释放它，也就是释放它占用的内存空间。要把这种思想贯穿到以后的编程中去，占用了内存，不用时要释放，切记！释放DC是一个很简单的函数： int ReleaseDC(HWND hWnd, // handle to window HDC hDC // handle to device context). 若成功释放，返回值是1，否则是0。参数有注释，我还是说一下： ※ HWND hWnd：你所要控制的那个窗口的句柄。如果你开始传递的是NULL，现在还要传递NULL。 ※ HDC hDC：DC的句柄。 在用DC和GDI进行图形显示前，我们先看看创建窗口实例时要遇到的几条重要的消息。我将要提到的四条消息是：WM_MOVE、WM_SIZE、WM_ACTIVATE、WM_PAINT。 追踪窗口状态 头两个是很简单的。当窗口被用户移动时将发送WM_MOVE消息，窗口新位置的坐标储存在lparam中。（还记得吗，消息在lparam和wparam中被进一步描述，它们是消息控制函数的参数）lparam的低端字中存储窗口客户区左上角的坐标x，高端字中存储坐标y。当窗口的大小被改变时，将发送WM_SIZE消息。同WM_MOVE消息差不多，lparam的低端字中存储客户区的宽度，高端字存储

高度。同WM_MOVE不同的是，wparam参数也控制了一些重要的东西。它可以是下列中任意一个值： ※ SIZE_MAXHIDE：其它的窗口被最大化了。 ※ SIZE_MAXIMIZED：本窗口被最大化了。 ※ SIZE_MAXSHOW：其它的窗口被还原了。 ※ SIZE_MINIMIZED：本窗口被最小化了。 ※ SIZE_RESTORED：窗口被改变了尺寸，但既没最大化，也没有最小化。 当我编写窗口实例时，我通常喜欢把窗口的当前位置和大小保留在几个全局变量里。假设我们命名这些全局变量为xPos，yPos，xSize和ySize，你最好这样控制WM_SIZE和WM_MOVE这两个消息： if (msg == WM_SIZE){ xSize = LOWORD(lparam). ySize = HIWORD(lparam).}if (msg == WM_MOVE){ xPos = LOWORD(lparam). yPos = HIWORD(lparam).}现在轮到WM_ACTIVATE消息了。它告诉你一个新窗口被激活。这是很有用的，因为如果出现优先的申请，你就不可能处理程序里的所有逻辑。有时，例如写一个全屏的DIRECTX程序，忽略WM_ACTIVATE消息将导致你的程序出现致命的错误，可能它做了一些你不希望它做的事情。在任何情况下，守候WM_ACTIVATE消息从而采取行动，是一个好主意。 窗口被激活和被解除激活都会发出WM_ACTIVATE消息，我们可以通过检测wparam的低端字来得知是被激活还是被取消。这将有三种可能的值： ※ WA_CLICKACTIVE：窗口被鼠标激活。 ※ WA_ACTIVE：窗口被其它东西激活。（键盘、函数调用、等等） ※ WA_INACTIVE：窗口被解除激活。 为了处理这个消息，我保留了另一个全局变量bFocus，当接收到WM_ACTIVATE消息，它的值将改变。示例如下： if (msg == WM_ACTIVATE){

if (LOWORD(wparam) == WA_INACTIVE) focus = FALSE. else focus = TRUE. // tell Windows we handled it return(0).}