游戏软件开发专家系列

DirectX 10 3D

游戏编程深度探索

（英）Peter Walsh 著
段菲 周飞 李威 译

清华大学出版社

游戏软件开发专家系列

DirectX 10 3D 游戏编程深度探索

(英) Peter Walsh 著

段 菲 周 飞 李 威 译

清华大学出版社

北 京

内 容 简 介

本书内容涵盖面广，讲解深入浅出，且示例丰富。书中主要介绍如何使用 DirectX 开发交互式 3D 图形程序，重点是游戏开发。全书首先介绍了必要的入门知识，如开发平台、图形学基础、数学工具，然后讲解了相关的 3D 概念。其他主题几乎涵盖了 Direct3D 中的所有基本运算，例如图元的绘制、光照、纹理、alpha 融合、模板，以及如何使用 Direct3D 实现游戏所需的技术。

本书可供从事 3D 游戏程序设计、可视化系统设计或其他图形应用程序开发的开发人员和大中专院校学生参考，也极适合各种游戏开发培训机构用作 Direct3D 编程的培训教程。

Advanced 3D Game Programming with DirectX 10.0 by Peter Walsh (ISBN 978-1-59822-054-4)

图书在版编目(CIP)数据

DirectX 10 3D 游戏编程深度探索/(英)沃尔什(Walsh, P.)著；段菲，周飞，李威译. —北京：清华大学出版社，2011.9
(游戏软件开发专家系列)
书名原文：Advanced 3D Game Programming with DirectX 10.0
ISBN 978-7-302-24920-7

Ⅰ.D… Ⅱ.①沃… ②段… ③周… ④李…Ⅲ. ①多媒体—软件工具，DirectX 10 ②游戏—应用程序—程序设计 Ⅳ. ①TP311.56 ②G899

中国版本图书馆 CIP 数据核字(2011)第 017937 号

责任编辑：文开琪 汤涌涛
装帧设计：杨玉兰
责任校对：周剑云
责任印制：李红英
出版发行：清华大学出版社 **地 址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn **邮 编**：100084
社 总 机：010-62770175 **邮 购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京密云胶印厂
装 订 者：北京市密云县京文制本装订厂
经 销：全国新华书店
开 本：185×230 **印 张**：33 **字 数**：741 千字
版 次：2011 年 9 月第 1 版 **印 次**：2011 年 9 月第1 次印刷
印 数：1～3500
定 价：79.00 元

产品编号：030637-01

前　言

历史上曾经有某位智者说过，以轶事作为一本书的开头是一种绝佳的方式。这句话的出处已无从考证，但对此，我一直深信不疑，因此本书也不例外。

当我还是一名高一新生时，我上了大多数同龄人都要上的必修课——生物课。这门课程要求我们完成一些试验、撰写实验报告以及解剖各种各样的小动物等很多诸如此类的事情。在我的实验室搭档中，有一个名叫 Chris V. 的小伙儿。我们俩都对计算机十分着迷，因此很快成为朋友，以至于在生物课上我们所谈论的主要话题都是一些“非正统”的技术问题。

12 月中旬的一个晚上，Chris 打电话给我。第二天就要上交的实验报告需要有来自于实验数据的结果，但他的那份实验数据找不到了。他想知道我是否可以帮他抄一份并将其带到他家，让他能够完成实验报告。当然，那时的我还处在无车可开的“悲惨世界”中，因此要驱车到 Chris 的住处必须先征得我父母的同意，另外还需要查清他的地址，总之很多困难等着我去克服。虽然我很乐意帮他的忙，但我并不希望毫无代价地伸手援助。因此我问 Chris 拿什么作为对我“仗义相助”的回报。

“噢，好吧，”他说，“我想我可以给你一份我刚刚得到的游戏拷贝。”

“真的吗？什么游戏？”我说。

“*Doom*[①]。是 Wolf 3D 的人制作的。”

“名字真的叫 *Doom*？这都是啥名字啊？”

抵达 Chris 家并将游戏拿到手之后，我们立即完成了“交易”，我迫不及待地将程序安装到我的古董级 386 DX-20 上。当我的星际战士“步履蹒跚”行进在地狱怪物层出不穷的走廊时，我的生活发生变化了。在念高中以前，我也编写过一些程序(主要是用 Logo 和 Basic)，但当我第一次体验过这个游戏之后，我发现自己对人生道路立即有了清晰的规划：我想编写像 *Doom* 这样的游戏！随后我访问了当地的几个公告板(BBS)，并提了两个问题：这个游戏是用什么语言编写的？使用的是什么编译器？

过了大概不到一天，我便购买了 Watcom C 10.0 以及我的第一本 C 语言编程书。我的第一个 C 程序自然是“Hello, World”。我的第二个 C 程序则是一个运行异常缓慢但令我无比兴奋的线框模型的旋转立方体。

我要向 John Carmack、John Romero 以及 *Doom* 团队的其他开发人员深深致敬：我对游戏开发的热情被他们的杰作彻底点燃了。也正是因为他们，我才学到了我目前所了解的关于这个充满无数乐趣和激动人

---

① 中文名为《毁灭战士》。

心的领域所需的编程知识。因此，我愿将自己多年的积累在本书中和盘托出，让那些对图形和游戏编程感兴趣的人少走一些弯路。

据我的个人体会，最大限度地在有限的篇幅内获取有用信息的最佳途径莫过于采用 FAQ(常见问题)模式，这一点早已无数次得到验证。我想当人们站在书店的书架前翻阅一本书时，一定希望能够尽快找到一些问题的解答，以便他们决定是否购买，或者接着读下去。

关于我的一些事儿

我是一名有着多年经验的游戏开发专业人员。我涉足这个行业的理由与当时大多数人一样，在 *Doom* 问世后，对游戏的工作机理产生了浓厚的兴趣。在自学编程后，我来到苏格兰的阿伯泰邓迪大学继续学习计算机游戏开发，并获得了一个学位。此后，我为一个名为 IC-CAVE 的下一代游戏开发技术的智囊团短期工作了一段时间。这些年，我先后参与了一些游戏的开发，如 *F1 Career Challenge*、*Harry Potter and Chamber of Secretes*、*SHOX* 以及 *Medal of Honor: Rising Sun*[①]。最近，我又参与了 Xbox 360 上的 *Crackdown*[②] 的开发工作。我几乎为所有您能够想到的平台开发过各种游戏。

此外，我还阅读过不计其数的编程书籍，我想应该有一半以上的亚马逊热带雨林为这些书做了“牺牲”。有了这些书籍的帮助，我希望自己能够避免其他作者不慎步入的所有“陷阱”。我真诚地希望您能够最大限度地从本书获益。如果您在学习过程中遇到难以解决的问题，请与我邮件(peter_stpwalsh@yahoo.co.uk)联系。很不幸，在本书的上一版出版后，我的邮箱便成了垃圾邮件经常光顾的地方，以至于我几乎将其弃用。不过，好在 Hotmail 越来越好，希望您的问题都能够被我看到。

写本书的初衷

我从许多非常有才华的人那里收获了许多，涉足了许多不同的领域，但也提过不少愚蠢的问题。我发现游戏开发产业的一个主流精神是分享。如果大家都乐于分享，则每个人都会了解更多，同时这个产业的知识总量也会不断增加。只凭一个人的力量是绝不可能发现计算机图形和游戏编程背后的所有真谛，也没有人能够只在真空中学习。从前，我从其他人那里分享到了很多，现在，该是我将自己所了解的知识与您分享的时候了。

注意： 自从升级到 DirectX 10 后，本书的所有源码都已更新为 Unicode 模式，并使用了安全的 STL。

本书面向的读者

本书专为那些已了解如何编程，但只是初涉或从未接触过图形和游戏编程的读者而写，例如那些来自

① 中文名为《F1 职业挑战赛》、《哈利·波特与密室》、《疯狂赛车》以及《荣誉勋章：日出》。
② 中文名为《镇压》或《除暴战警》。

其他领域的开发人员或是希望从事一些兼职项目的大学生。

不适合阅读本书的读者

本书面向的读者并不是初学者。我并非在此故弄玄虚，我当然相信如果初学者感觉良好，他也许可以将本书啃下来。但限于篇幅，我不得不对一些重要但非本书重点的概念(如 C++的继承机制)的讲解进行压缩。如果您以前从未接触过编程，阅读本书时，您将遇到极大的困难。

本书配套代码的运行环境

本书的所有源码都是在 Microsoft Visual Studio 2005 Express 环境中使用 C++编写的，前者可从 Microsoft 的官网免费获取。每个项目(project)和解决方案(solution)都可从本书的配套站点(www.wordware.com/files/dx10)下载得到。很显然，本书只关注 DirectX 10，因此在我编写书中代码时，它们只能运行于 Windows Vista 或更高版本的 Windows 上，且显卡必须支持 DirectX 10。如果尚不具备这些条件，需要先进行升级。

为何使用 Windows 而非 Linux

我写代码时，选用了 Win32 API 环境，这是因为目前大约有 90%的计算机用户的工作环境都是 Windows。Win32 API 并不是一种易于理解的 API，尤其是对那些熟悉 DOS 编程约定的开发人员而言。Win32 API 也并不优美，但我想它也不至于一无是处。我本可以选择其他平台，但这样做只会导致本书的读者减少九成甚至更多。

为何使用 Direct3D 10

如果您之前从未使用过 Direct3D 10，您可能听说过另一种图形 API——OpenGL。这种 API 于 20 世纪 90 年代由 Silicon Graphics 公司设计，并用于其高端图形工作站中。随后，这套 API 被移植到不计其数的平台和操作系统中。除游戏产业外，在其他领域(如仿真和学术界)，OpenGL 已成为实现图形学的事实上的标准。OpenGL 简单易用，十分优雅，同时非常高效。您可从 www.opengl.org 了解到更多有关 OpenGL 的信息。

OpenGL 固然有许多优点，但它并不完美。首先，OpenGL 拥有数量极其丰富的函数。为了使接口尽可能简单，实现中不得不考虑许多繁缛的细节，以确保程序的正常运行。为与 OpenGL 驱动完全兼容，每个 3D 显卡制造商都必须支持完整的 OpenGL 特性集。要想完全准确地实现这些驱动异常困难，而且相同的显卡所表现的性能可能因驱动程序的质量而发生很大的差异。而 DirectX 与 OpenGL 相比的一个优势是它总能迅速更新，并与最新的硬件保持兼容。DirectX 为 Microsoft 所控制(也许是好事，也许是坏事，仁者见仁)，而 OpenGL 扩展则需要 OpenGL 体系结构委员会的慎重讨论。这样产生的一个后果就是，如果

使用 OpenGL，将无法享受到 DirectX 已提供的最新的对着色器的支持。

为何使用 C++而非 C、.NET 或 Java

在编写本书时，我曾经考虑过是否采用几种其他的语言。虽然 Delphi、VB 甚至 C#都有一些非常突出的优点，但我真正严肃考虑过的语言只有 C++、Java 和 C。Java 由 Sun Microsystems 公司所设计，是一种完全面向对象的语言，且具有一些高级特性，如垃圾回收。C 则是一种接近底层，却不需触及汇编语言的高级编程语言。C 语言只有极其少量的高层结构，做的抽象也非常少。

C++是一种非常有趣的语言，因为它的功用恰好介于前述的两种语言之间。C++对 COM 的支持与 C 相比更好(在第 1 章中有比较详细的讨论)。它的类系统和运算符重载功能增强了代码的可读性(当然任何强大的功能都可能被滥用)。Java 虽然非常酷，但它是一种解释性的语言。随着时间的推移，Java 的这个缺点越来越微不足道：JIT 编译的速度越来越快，同时越来越多的耗时工作也移交给 API。尽管如此，我个人感觉 C++更适合本书。Java 毕竟是一门年轻的语言，必将经历许多蜕变。

是否需要 3D 加速器

是的。并且它必须支持 DirectX 10，并运行在 Windows Vista 或 Windows 7 平台上。

本书对读者的 C++基础要求

C++在一些人眼中被视为尚方宝剑。他们以一种闻所未闻的方式控制着模板类。他们为自己编写的所有类都重载了输入/输出流运算符。他们将多重继承视为地狱中的恶魔。在我的眼中，C++只是一个工具。在我的工作中，对 C++的一些过于复杂的特性(如 iostream 库)我几乎从不使用。而那些相对不太复杂的特性(如多重继承)，我只是在必要的时候才会使用。坚持一种良好的编码风格非常重要。本书所有代码的编写持续了 11 个月，外加 3 个月的修订时间。但是当我拿出最初编写的那些代码时，我依然可以轻松地读懂它们，这是因为我始终保持添加注释的习惯，并遵循一些优良的编程约定。希望您也能像我一样，或做得比我更好。

本书中源代码的编程约定

在我所阅读过的编程书籍之中，最棒的一本当属 Microsoft Press 出版的 *Code Complete*(中文名为《代码大全》)。该书对变量名的长度、子函数的设计、文件长度等细节问题做了详尽的讨论。我强烈建议那些希望成为伟大程序员的朋友仔细研读该书。您可能会注意到我在本书中使用的一些约定与 *Code Complete* 中的非常类似。需要说明的是，其中一些借鉴自若干伟大的游戏开发人员如 John Carmack，而另外一些则借鉴自 DirectX 和 Win32 的源码。

我尽了自己最大的努力以使本书的代码能够为每一位读者所理解。我在所有自己认为可能引起疑惑的地方都添加了注释，所有的变量名我都经过了仔细斟酌，并尽力使代码做到简洁而高效。当然，众口难调，这些努力不可能取悦所有的读者。我相信有一些C++编码标准我可能并没有正确地遵循，而且一定有些代码片段并不是最高效的。

如果您以前从未使用过 C++或刚接触编程，您的阅读之旅一定会充满艰辛。对这类读者，我想推荐 Lippman 编著的 *C++ Primer*(Addison-Wesley 出版)，这本书无论对编程的精髓还是 C++都有非常深入的讨论。

类名和结构名

在本书的代码中，所有的类名都以小写字母 c 开头，所有结构体均以小写字母 s 开头，所有接口名前都以小写字母 i 开头，而所有枚举名前均以小写字母 e 开头。

当然有一些例外情况需要注意。虽然大多数类在设计时都试图做到功能隐藏，并以组件的形式加以利用，但也有少量类和结构作为基本类型而设计。因此对于一些基本的数学元素如点和矩阵，我并未添加前缀，而是在其后加上表示该类型维数的后缀(如 2D 点结构命名为 point2，3D 点结构命名为 point3，依此类推)。这样做的用意是使它们与其概念近邻 float 保持一致的风格。出于同样的考虑，所有的数学基本类型都重载了大量运算符以简化操作。

变量名

中等长度的变量名是非常值得推荐的。这种命名习惯会使您的代码见名知意。如果变量名称过长，就会使代码整体显得很不和谐，同时也会延长代码的编写时间。

我偶尔也会使用一些非常简短的变量，“int i, j, k”这样的变量在循环结构中大量被使用。但在其他场合，我都尽量为我所使用的每一个变量赋予有意义的名称。这意味着这些有意义的变量名中一般都包含了至少一个单词。在我的变量命名系统中，组成变量名称的第一个单词的首字母都为小写，而随后单词的首字母均为大写，单词与单词之间没有下划线，例如 int numObjects。如果一个单词的最后一个字母是大写字母，则需附加一个下划线以与随后的单词分隔开来，例如 class cD3D_App。

有一种流行的命名方法称为匈牙利命名法，我们将在第 1 章中进行介绍。我并没有严格遵循该命名方法，但我通常会为浮点型变量名加上前缀 f，在整型变量前加上前缀 i，而在指针类型前加上前缀 p(例如：float fTimer; int iStringSize; char * pBuffer)。请注意这里的前缀与首单词类似，其后的所有单词的首字母都为大写。(因为我发觉 pBuffer 通常比 pbuffer 具有更好的可读性。)

对于一些特定性质的变量，我也为其增加了前缀。所有的全局变量一般都以“g_”开头(如 g_hInstance)，所有的静态变量均以“s_”开头(如 static float s_fTimer)，而所有的类成员变量均以“m_”开头(如 int m_iNumElements)。

配套文件

本书的所有配套文件均可从如下网址获取：

www.wordware.com/files/dx10[①]

这些配套文件中包含本书中所有例程的源代码，并附送游戏“Mobots Attack!”。每章的例程均拥有独立的解决方案，以方便您独立使用。

① 编者注：至本书中文版出版之时，该网址已更改为 http://www.jblearning.com/Catalog/9781598220544/student/。读者还可从清华大学出版社网站下载配套文件，地址为 http://www.tup.com.cn 或 http://www.wenyuan.com.cn。

目　录

第 1 章　关于 Windows

大家好，欢迎开始 DirectX10 3D 游戏高级开发的学习之旅，这里将是我们深入学习的起点。要学好 3D 游戏编程，建立完善的知识体系，首要任务是了解与 Windows 有关的基本知识。我们将先用较短的时间来学习一些简单的操作，如程序的打开和关闭、基本输入的处理、基本图元的绘制等。如果您对 Windows API 已经很熟悉，在阅读本章内容时将倍感轻松；否则，请仔细阅读本章！

本章主题：

- Windows 的一些理论知识以及如何使用 Win32 API 进行程序开发
- Win32 游戏开发和标准的 Windows 编程有何不同
- 消息及消息处理
- 标准消息泵和实时消息泵
- Win32 编程
- COM，即组件对象模型(Component Object Model)
- 其他更丰富的内容

1.1　关于 Windows

经过 15 年的发展，Windows 拥有一套非常标准的 API，即应用程序编程接口(Application Programming Interface)。1992 年，我们为 Windows 3.11 操作系统所编写的程序，理所当然地仍旧可以在 Windows Vista 下运行。所以，我们在本章所学习的内容作为 Windows 编程标准已经存在十多年了。本书的所有代码都是基于 DirectX 10 的，因此仅能运行于 Windows Vista 及后续版本的操作系统，也就是说 Windows Vista 是保证本书代码能够正常运行的最低配置的操作系统。当然，绝大多数内容只需稍加修改就能适用于之前版本的 Windows 和 DirectX。

现在的 Windows 程序同古老的 DOS 程序或 Win16 程序相比，在很多方面都有着根本性的变革。在过去，运行一个程序将占用 100%的 CPU 时间，且程序员对所有的设备和文件有 100%的控制权。因此您还必须熟知用户机器中的设备的诸多细节(您可能还记得那些神秘的 DOS 或 Win16 游戏，它们几乎总是要求您输入某些设备的 DMA 和 IRQ 配置，如声卡的配置)。这样当游戏出错时，整个系统都会随之崩溃。对于终端用户而言，这意味着只能选择重新启动机器。

谢天谢地，在 Windows Vista 和 DirectX 10 中，上述情形完全不会出现。当您的程序执行时，它会和许多其他进程一起共享 CPU，并发(在同一时间)地运行。因此，您无法对声卡、显卡、硬盘以及其他所有

的系统资源施以完全的控制。输入、输出也已被抽象出来。

这无疑是一件非常好的事，但这也需要您在最开始时多花费一些学习时间。

一方面，Windows 应用程序都具有一致的界面(look and feel)。所以，几乎所有开发人员创建的 Windows 应用程序都自动地被 Windows 用户所熟悉。他们早就了解如何使用菜单和工具栏，因此，如果使用 Windows 的基本框架来构建应用程序，用户很快就知道如何使用这个用户界面。

另一方面，应该对 Windows 和其他应用程序抱有足够的信心。在 DirectX 出现之前，我们只能使用一些 Windows 默认的绘图命令(被称为 GDI 或图形设备接口)来绘图。虽然 GDI 可以自动处理任意位深度的图像，并且能在任何监视器上工作，但是效率非常低。Windows Vista 采用了全新的显示驱动模型，该模型完全重写了使用 DirectX 的用户接口。经过这种改进后，效率比以往的 Windows 版本快了许多，但对于格斗游戏来说仍不是特别理想。由于这个原因，很多 DOS 程序开发人员发誓不在 Windows 上工作。在渲染复杂场景时，最好的方案是将场景绘制到一幅位图中，然后再将该位图呈现到某一窗口中，这是一个非常缓慢的过程。编写 Windows 应用程序时，我们必须放弃许多东西。

然而，很多在 Windows 中能做到的事情，对于 DOS 环境来说简直就是个噩梦。在 Windows 编程中，我们可以仅使用一行简单的代码(PlaySound 函数)来实现音效，查询时间戳计数器，使用健壮的 TCP/IP 网络栈，访问虚拟内存等。虽然需要花点时间到处找这些功能函数，但是 Windows 的优势远远胜过它的劣势。

本书所有应用程序的代码都是在 Windows Vista 环境下用 Visual C++ 2005 速成版(可从微软官方网站免费下载)编写的。我们将使用 C++编写游戏程序，并且还要用到 Windows SDL(Simple DirectMedia Layer)平台下的 Win32 API。Win32 API 就是一组 C 语言函数，开发人员可以使用这组函数来构建依赖于 Windows 平台的应用程序。它把一些如多任务和内存保护一类的复杂操作都抽象出来，并且向用户提供高层概念的接口。此外，Win32 API 还提供了对菜单、对话框和多媒体的支持。

我们可以把 Windows 看成一套非常丰富的 API 集合。从播放视频到加载网页，您可以无所不为。而且，对于每一项任务，都可以有很多种不同的方法去实现它。目前，有很多大部头的著作都着眼于讲解 Windows 编程的更基础的概念，我们则只关注本书用得着的知识。因此，对于诸如创建带树控件的对话框、打印文档、读写注册表中的键值之类的大段知识，我们是不会花时间来学习的。我们需要处理的只是最简单的情况：创建一个能够用来绘图的窗口，并把输入数据传递给程序，这样我们至少已经开始了和操作系统之间的良性互动。如果您需要了解更多 Windows 编程的知识，有很多资源可以供您学习。我特别推荐的是 Charles Petzold 的《Windows 程序设计(第 5 版珍藏版)》。

1.2 匈牙利命名法

在 Windows 领域中对变量进行命名一般都采用所谓的匈牙利命名法。这个名字源自它的发明者 Charles Simonyi，他是个匈牙利人。

匈牙利命名法(Hungarian Notation)是一种编码约定，就是在变量名称前面加上一些前缀字母，用来标识这些变量的类型。有了匈牙利命名法，我们阅读其他人的代码就变得更加容易，并且更能保证给函数传递的是正确格式的变量。不过，对于没有接触过匈牙利命名法的人来说，可能还是会感到一些困惑。

表 1.1 给出了一些常用前缀，它们在本书的绝大多数代码中会被用到。

表 1.1　常用的匈牙利符号前缀

常用前缀	示　例	说　明
b	bActive	BOOL 类型的变量，它是 C++中布尔类型的前身。 其值可以是 true 或者 false。
l	lPitch	长整型变量。
dw	dwWidth	DWORD 类型或者是无符号长整型变量。
w	wSize	WORD 类型或者是无符号整型变量。
sz	szWindowClass	指向字符串的指针变量，该字符串以 0 结束(即标准的 C 风格字符串)。
p 或 lp	lpData	指针变量(lp 是由以往的远指针演化来的，也就是长指针，即 long pointer)。一个指向指针的指针用 pp 或者 lplp 来表示，依此类推。
h	hInstance	Windows 句柄变量。

1.3　Windows 的一般概念

记事本程序可以说是 Windows 简单程序的一个很好的范例。它允许基本文本输入，能够让您进行基本文字处理，如查找和使用剪贴板，同时通过它您也可以打开、保存和打印一个文件。该程序如图 1.1 所示。

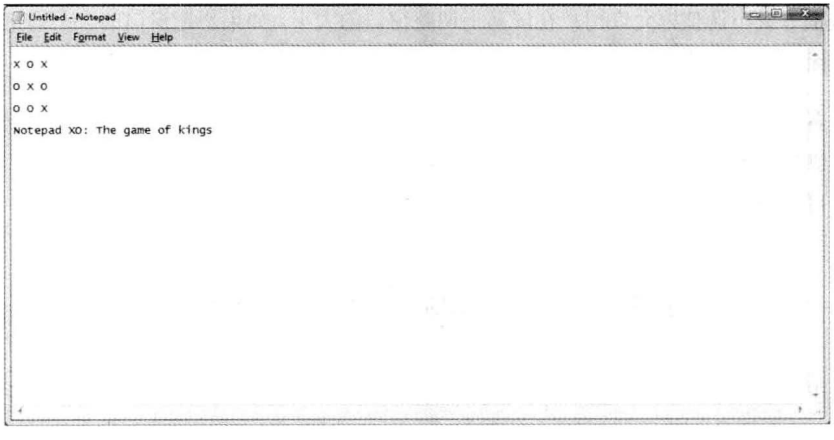

图 1.1　记事本——最基本的窗口

我要教您创建的窗口和这个窗口很类似。这样的窗口可被划分成几个不同的区域(如图 1.2 所示)。Windows 会管理其中的一些区域，剩下部分的就需要您的程序来管理。

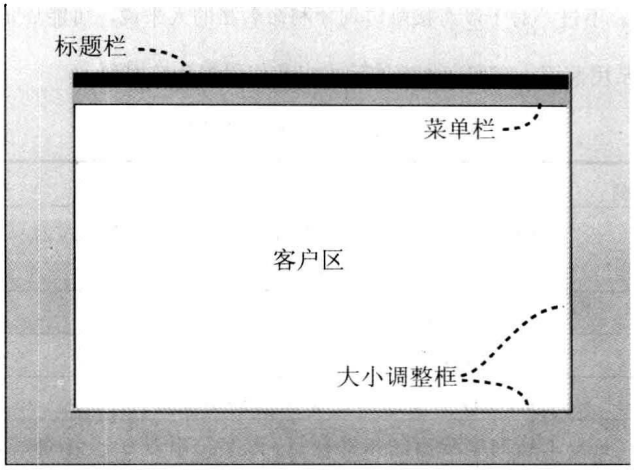

图 1.2 GUI 窗口组件

窗口的主要部分如下所示。

标题栏 该区域在大多数窗口中都会有。它给出了窗口的名字并且提供系统按钮，如关闭、最小化或者最大化应用程序。您可以在创建窗口的过程中通过一些标记来控制标题栏，如可以让它消失、不带系统图标显示或者更窄一些。

菜单栏 菜单是 GUI 程序中最主要的一种交互形式。它提供了命令列表，用户可以在任何时候执行这些命令。当然，您也只需要创建菜单并且定义好命令，其他的事情就交给 Windows 来处理。

大小调整框 大小调整框允许用户在屏幕上调整窗口的大小。如果想把窗口的大小固定，可以在创建窗口的时候关闭相应的选项。

客户区 客户区是您所要处理的区域。Windows 实际上在该区域为您提供了一个用于操作的沙箱。这里就也是您绘制场景的地方。Windows 也可以在这个区域的某些部分进行绘图。当应用程序中带有滚动条或工具条时，可以说它们占用了客户区的一部分。

1.4 Windows 中的消息处理

Windows 中有个概念，叫焦点(focus)。在同一时刻只能有一个窗口获得焦点。只有获得焦点的窗口才能与用户进行交互，而其余的窗口都只相当于背景，并且标题栏颜色与当前的窗口不同。因此，只有一个应用程序可以获知键盘的当前状态。