游戏软件开发专家系列

Direct 3D和XNA游戏开发基础(C#语言版)

耿肇英 编著

XNA + C#=新一代游戏开发利器

★ 采用实例教学法，在讲清基本知识点的基础上，借助于丰富的实例加以说明

★ 涵盖关健知识点，提供短小精悍的范例代码，并辅之以详细设计步骤和解释

清华大学出版社

游戏软件开发专家系列

Direct 3D 和 XNA 游戏开发基础(C#语言版)

耿肇英　编著

清华大学出版社

北　京

内 容 简 介

使用 C#语言可以编写基于 DirectX 和微软最新游戏开发平台 XNA(仅支持 C#语言)的 3D 图形和 3D 游戏程序，其运行速度已接近于 C++代码的运行速度，一些商业游戏已经使用 C#语言创建。用 C#语言编写 3D 图形和 3D 游戏程序可以降低学习和开发难度，提高开发效率，使开发人员能写出更安全的代码。本书目的是使具有 C 语言基础的读者通过本书学习，掌握用 C#语言开发基于 DirectX 和 XNA 的 3D 图形和 3D 游戏程序。本书采用实例驱动的方式进行讲解，在例子中尽量避免罗列不相关的知识点和无关代码，使例子代码短小精悍，容易理解，书中全部例程均给出了详细设计步骤，并对每一步代码给出详细解释，读者可按照书中步骤完成例子。

本书可作为学习用 C#语言开发 3D 图形和 3D 游戏程序的入门书，也可作为高校计算机及游戏等相关专业教师、研究生、本专科学生的教材或参考书，对使用 C#语言开发 3D 图形和 3D 游戏程序的程序员也有很好的参考价值。

图书在版编目(CIP)数据

Direct 3D 和 XNA 游戏开发基础(C#语言版)/耿肇英编著. —北京：清华大学出版社，2009.1
(游戏软件开发专家系列)
ISBN 978-7-302-18764-6

Ⅰ.D… Ⅱ. 耿… Ⅲ. ①多媒体—软件工具，Direct ②游戏—应用程序—程序设计 Ⅳ. TP311.56 G899

中国版本图书馆 CIP 数据核字(2008)第 161811 号

责任编辑：文开琪
封面设计：杨玉兰
责任校对：李玉萍
责任印制：

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：
装 订 者：
经　　销：全国新华书店
开　　本：185×230　　**印　张：**24.75　　**字　数：**533 千字
版　　次：2009 年 1 月第 1 版　　**印　次：**2009 年 1 月第 1 次印刷
印　　数：1~4000
定　　价：46.00 元

前　言

运行于 Windows 操作系统的大部分 3D 游戏和图形程序都是基于微软公司 DirectX 开发的，以往开发 DirectX 程序主要使用 C++语言。由于 DirectX 函数库的使用十分复杂，所涉及的概念众多，而且很多都不容易理解，又加上 C++语言学习难度较大，以及国内适合初学者的有关 DirectX 程序设计书籍较少等原因，使用 DirectX 设计 3D 游戏和图形程序工作一直未能得到很好的普及。

2002 年微软推出 Managed DirectX(托管 DirectX)，支持 C#语言开发 DirectX 程序。2006 年 8 月微软发布 XNA Game Studio Express，是微软专门用于开发游戏的最新集成开发环境，是微软大力发展的游戏开发平台。Managed DirectX 和 XNA 用类重新封装了 DirectX 函数库，比直接使用 DirectX 函数库要容易得多。两者都使用 C#语言开发，减少了学习编程语言的难度。使用 C#语言编写基于 Managed DirectX 和 XNA 的 3D 游戏和图形程序可以降低学习和开发难度，提高开发效率，使开发人员能写出更安全的代码。基于 Managed DirectX 和 XNA 的 3D 游戏和图形程序的运行速度已很接近 C++程序，一些商业游戏已经使用 Managed DirectX 和 XNA 来创建。XNA 编写的 3D 游戏程序还可运行于微软游戏机 XBox 360。Managed DirectX 和 XNA 的推出必将促进 3D 游戏和图形程序设计的普及。

本书介绍使用 C#语言开发基于 Managed DirectX 和 XNA 的 3D 游戏和图形程序技术，目的是使具有 C 语言基础的读者通过本书学习，掌握使用 C#语言开发 3D 图形程序的基础，并能开发简单的 3D 游戏程序。

本书采用实例教学法，在讲清基本知识点的基础上，尽量使用实例加以说明，因此书中包含了大量实用例子，在例子中尽量避免罗列不相关的知识点和无关代码，使例子代码短小精悍，容易理解。书中绝大部分例子都给出了详细设计步骤，并对每一步的代码给出详细解释，读者可按照书中步骤实现所有例子。本书所有例子源代码可以从清华大学出版社网站(*www.tup.com.cn* 和 *www.wenyuan.com.cn*)下载。

本书分为两部分，前一部分主要讲解 Managed DirectX 3D 1.1 程序设计基础，介绍 Direct 3D 的一些基本概念，由于 XNA 仍以 DirectX 9.0 C 为基础，因此这些概念也完全适应于 XNA。本书后半部分介绍 XNA 中的基本概念和框架，用大量的篇幅说明从 Managed DirectX 3D 1.1 移植到 XNA 的方法，并移植了前半部分的大部分例子，同时也介绍了一些 XNA 所特有的概念和方法，并使用这些概念和方法实现了一些例子。本书的最后一章是一个用 XNA 实现的简化版赛车游戏实例，给出了详细的设计步骤，学完本书的读者都可以完成。

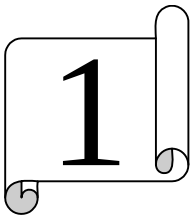
本书适合以下几类读者。

- 第一类是游戏专业的学生。由于现在游戏公司一般还是使用 C++和 DirectX 编写游戏，所以游戏专业的学生主要还要学习 DirectX，但同时也应该学习 XNA。学习 Managed DirectX 相对比较容易，因此可以把本书作为学习 DirectX 前导课程的教材。同时兼顾 XNA 技术和 Managed DirectX 概念，可以为以后学习 DirectX 打好基础。
- 第二类是正在学习或者已基本掌握使用 Managed DirectX 1.1 来编写游戏的程序员，他们希望学习 XNA。通过本书的学习，可以很快掌握 XNA。
- 第三类是用 C++和 DirectX 开发游戏的程序员。由于 XNA 是游戏开发的方向，但 XNA 不支持 C++语言，仅支持 C#语言，因此这类读者应该有相当多的人要学习 XNA。由于 Managed DirectX 和 DirectX 在概念上有对应关系，所以这些读者学习 XNA 的捷径应该是首先学习 C#语言，利用以前对 DirectX 概念的理解来编写 Managed DirectX 3D 程序，编写中不会遇到 3D 概念上的问题，然后轻松从 Managed DirectX 转向 XNA。使用本书可以使 C++程序员更快地掌握 XNA。
- 第四类是希望在 C#语言 Windows 应用程序中增加一些 3D 功能的读者由于在程序中仍然要使用菜单、工具条、对话框、按钮等 WinForm 控件，Managed DirectX 3D 1.1 足以应付这些要求，因此这本书也非常适合这些读者。虽然微软表示不再升级，但会继续支持 Managed DirectX 3D 1.1，因此其生命周期不会马上结束，即使需要升级到 XNA，本书也为这些读者做好了准备。
- 最后一类是初学者。他们以前从没有接触过 Direct 3D，学习的目的仅仅是使用 XNA 编写游戏程序，这些读者的确应该从一开始就学习 XNA，采用这本书实际走了一个弯路。如果选不到其他合适的书，也可选用本书。作者对这类读者有一个学习建议。第 2 章必须掌握。第 3 章需要掌握 Device 类的概念，基本框架要会使用，无论如何也必须使用本框架做一些例子，要掌握 VertexBuffer 类和背面剔除概念，要会使用 DrawUserPrimitives 方法，本章例子则不必做。第 4 章的世界、观察和投影变换是最重要的概念之一，建议动手做例 4.1、例 4.2、例 4.6 和例 4.9。第 5 章重点掌握灯光、材质和法线的概念，建议只看前 4 节，动手做做例 5.1。第 6 章可以只看前 5 节及例子。第 7 章看一下前 2 节，XNA 中没有 Mesh 类。看完本章后也可以直接看第 10 章，这章是必须学习的，因为 XNA 仅支持可编程流水线。后续各章都是讲解 XNA，可顺序学习，学习中遇到概念问题，可参见前面的相应章节。请注意，第 8 章和第 12 章以及第 9 章和第 13 章是对应的，第 8 章和第 9 章例子使用 Managed Direct 3D 1.1 实现，第 12 章和第 13 章的例子使用 XNA 实现，可以采用第 8 章和第 9 章中的概念，做第 12 章和第 13 章的例子，通过这种方式来学习。

本书主要由耿肇英完成，耿燚、杜伟浩、杜建文、周真真、姜志敏、霍利岭、胡春叶、董雪梅、张艳格也参加了本书部分代码的编写、调试、整理及录入校对等工作，在此一并表示感谢。在这里还要特别感谢文开琪编辑对本书出版的大力支持，也要感谢段菲博士对本书提出的中肯的意见。

由于时间仓促，加之水平有限，书中的缺点和不足之处在所难免，敬请读者批评指正。联系方法：gengzhaoying@sina.com。

目　录



Managed DirectX 和 XNA

本章首先介绍 DirectX、Managed DirectX 和 XNA 有关知识，然后详细解释用 C#语言设计 3D 程序时所用到但一般教科书又很少讲到的重要概念，最后列出 C#语言 3D 程序设计中用到的一些结构。本书认为读者已经掌握了 C#语言基本语法，并能够使用微软 Visual C# 2005 速成版(简称 VS 2005)编写 Windows 应用程序。如果读者不具备这方面的技能，请先阅读附录，掌握最基本的 C#语言语法，然后再学习本章，学习事件驱动有关概念和用 VS 2005 编写 Windows 应用程序的最基本步骤，为学习用 C#语言编写 Direct 3D 和 XNA 程序提供必要的基础。当然，仅靠这些内容就完全掌握 C#语言是不可能的，如需进一步学习 C#语言，还需要阅读 C#语言的专著。如果读者对这方面的内容比较熟悉，也可跳过本章 1.4 节以后的内容，直接阅读第 2 章。

1.1 DirectX

DirectX 是微软公司推出的运行于 Windows 操作系统的多媒体 API，其中 Direct 3D 支持 2D、3D 图形程序开发，可用来设计 2D 和 3D 游戏程序，已成为在 Windows 操作系统下 3D 游戏程序设计的事实标准，是开发运行于 Windows 操作系统的 3D 图形和游戏程序必须掌握的技术。最新微软游戏开发平台 XNA 也以 DirectX 9.0 C 为基础。

当前开发 DirectX 程序主要使用 C++语言，为了使用 DirectX 进行程序设计，必须首先到微软网站免费下载并安装 DirectX SDK。如果运行环境为 Windows 2000，请安装 Direct X SDK 9.0 C，其后续的版本不支持这个操作系统。

1.2 Managed DirectX

以往开发 DirectX 程序主要使用 C++语言，相对于 Visual Basic 语言，C++语言学习难

度较大。而 DirectX 概念众多，许多概念又不容易理解，又加上国内适合初学者的有关 DirectX 程序设计书籍较少，很多学习 DirectX 程序设计的读者往往付出了很大努力，最后不得不半途而废。

C#语言是一种现代的、面向对象的语言，它简化了 C++语言在类、命名空间、方法重载和异常处理等方面的操作，它摒弃了 C++的复杂性，更易使用，更少出错。它使用组件编程，和 VB 一样容易使用。学习 C#语言要比学习 C++语言容易得多。

2002 年微软推出 Managed DirectX，也称作 DirectX 托管版本，用类重新封装了 DirectX 函数库，支持 C#和 VB.NET 语言开发 DirectX 程序，极大地简化了 DirectX 程序设计。

一般认为，和 C++语言相比，C#语言程序运行速度要慢很多。实际上经过测试，C#语言 3D 程序运行速度已接近于 C++语言 3D 程序的运行速度，因此一些商业游戏已经使用 C#进行创建。虽然某些性能要求特别高的特定游戏可能需要使用 C++，但大多数游戏可以使用 C#语言创建或同时使用 C#和 C++语言创建。

使用 C#语言编写 3D 图形和游戏程序可以降低学习和开发难度，提高开发效率，使开发人员能写出更安全的代码。本书前半部分介绍使用 C#语言开发基于 Managed DirectX 的程序，目的是使读者通过本书的学习，掌握使用 C#语言开发 3D 图形程序的基础。

为了使用 Managed DirectX 进行程序设计，必须首先到微软网站免费下载、安装 DirectX SDK 9.0，该版本也支持 Managed DirectX。本书所有例子的调试和运行环境为微软 Windows XP 中文专业版，VS 2005，.Net FrameWork 2.0，DirectX SDK 9.0C(April 2005)。安装 DirectX 比较容易，一般情况下先安装 VS 2005，然后安装 DirectX SDK 9.0C，安装时会自动对 VS 2005 进行必要配置，以方便使用。Managed DirectX SDK 9.0C 包含 13 个命名空间，其定义的命名空间及其主要用途如下。

- **Microsoft.DirectX** 包含公共类和数学结构，例如向量和矩阵。
- **Microsoft.DirectX.Direct 3D** 用于开发 3D 图形和 3D 游戏。
- **Microsoft.DirectX.DirectDraw** 2D 图形 API。Direct 3D 中已包含 2D 图形功能，所以新的应用程序不应再使用它。
- **Microsoft.DirectX.DirectPlay** 用于开发多人网络游戏。
- **Microsoft.DirectX.DirectPlay.Lobby** DirectPlay 扩展，支持 Client/Server 型游戏。
- **Microsoft.DirectX.DirectPlay.Voice** 在应用程序中添加声音特性。
- **Microsoft.DirectX.DirectSound** 声音支持。
- **Microsoft.DirectX.DirectInput** 提供输入设备支持(例如，鼠标和游戏杆)。
- **Microsoft.DirectX.AudioVideoPlayback** 播放视频和音频文件。
- **Microsoft.DirectX.Direct3DX** 包含许多创建 Direct 3D 应用程序时用到的常见函数。

- **Microsoft.DirectX.Security**　访问安全性。
- **Microsoft.DirectX.Security.Permissions**　确立安全行为和安全规则，访问权限。
- **Microsoft.DirectX.Diagnostics**　用来检测环境特性。

1.3　XNA 简 介

微软于 2006 年 8 月发布 XNA Game Studio Express(简称 XNA)，是微软开发游戏的最新集成开发环境的简化版，仅支持 C#语言，可以免费下载。XNA 的第一个目的就是使游戏编程变得更容易，将检查显卡、创建 Device 设备、消息事件处理、纹理导入、x 和 fx 文件导入等工作都交由 XNA 完成，程序员的工作就是编写游戏逻辑代码。XNA 的第二个目的是使用 XNA 编写的跨平台游戏可同时运行于微软操作系统和微软游戏机 Xbox 360。XNA 和 Managed DirectX 不兼容，最主要的区别是不支持固定功能流水线，仅支持可编程流水线；不再采用左手系统，而采用右手系统；不包括 Managed DirectX 中的一些类。但它仍然以 DirectX 9.0 C 为基础，和 Managed DirectX 有许多共同点，两者是相通的，有很好的相容性。掌握 Managed DirectX 后，可以很快掌握 XNA 框架。XNA 完全可以代替 Managed DirectX 1.1 3D 游戏程序框架，使用起来更加简单，是微软大力发展的游戏开发平台，是游戏编程的方向。已有 XNA 2.0 版。本书后半部分介绍使用 C#语言开发基于 XNA 的游戏程序，目的是使具有 C 语言基础的读者可通过本书的学习，掌握使用 XNA 开发 3D 游戏的基础知识。如何安装 XNA 请参见 11.1 节。

1.4　.NET Framework

.NET Framework 是使用 C#编写 Windows 应用程序的基础类库。Windows 98、2000 和 XP 操作系统不包含这个类库，为了运行 C#程序，必须安装.NET Framework。.NET Framework 目前有 1.0、1.1 和 2.0、3.0、3.5 等几个版本。本书使用的是 2.0 版本。

1.5　事 件 驱 动

事件是 C#语言内置的语法。Windows 应用程序采用事件驱动方式工作，也叫消息驱动。Windows 操作系统是一个多任务的操作系统，允许同时运行多个程序，它不允许任何一个程序独占外设，如键盘、鼠标等，所有运行程序共享外设和 CPU，各个运行程序都要随时准备从外设接受命令，执行命令。因此必须由 Windows 操作系统统一管理各种外设。Windows 把用户对外设的动作都看作事件(消息)，如单击鼠标左键，发送单击鼠标左键事

件，用户按下键盘，发送键盘被按下的事件等。Windows 操作系统负责管理所有的事件，根据具体情况把事件发送到相应运行程序，而各个运行程序自动用一个函数响应事件，这个函数叫事件处理函数。Windows 应用程序一般总是在等待事件发生，一旦接到 Windows 操作系统发来的事件，立刻调用事件处理函数处理事件，处理完成后，将再一次进入等待状态，这种工作方式叫事件驱动，这种工作方式的最大优点是能够充分利用 CPU。

C#语言可使用组件来编写 Windows 应用程序。组件本质上是类。在组件类中，可预先定义该组件能够响应的事件，该事件发生，程序员编写的相应事件处理函数便被自动调用。例如，按钮类定义了单击事件 Click，单击按钮就会调用程序员编写的该事件的事件函数。一个组件可能定义了多个事件，应用程序中没必要响应所有的事件，而只需响应其中部分的事件。现在的问题是，如何把程序员编制的事件处理函数和组件类中预先定义的事件联系起来。以按钮单击事件 Click 为例，为单击事件 Click 指定事件处理函数的代码如下：

```
//指定 button1_Click 函数是按钮单击事件 Click 的单击事件处理函数,参见例 1.2 中的代码
button1.Click += new System.EventHandler(button1_Click);
private void button1_Click(object sender, EventArgs e)  //Click 事件处理函数
{    this.button1.Text = "单击了我";   }                 //发生 Click 事件自动调用本函数
```

在使用 Managed DirectX 编写程序时，程序员经常要用 C#语言代码将事件和事件处理函数联系到一起，而不是由 VS 2005 集成环境自动添加代码，为事件指定事件处理函数。请读者务必学会用程序代码为事件指定事件处理函数。

基于 Managed DirectX 和 XNA 的 3D 图形和游戏程序也采用事件驱动方式工作。和 Windows 应用程序不同的是，在空闲时间，基于 Managed DirectX 和 XNA 的 3D 图形和游戏程序总是循环调用渲染函数，在窗口绘制图形，而不是处于休眠状态，换言之，基于 Managed DirectX 和 XNA 的 3D 图形和游戏程序绘制图形不采用事件驱动方式工作，除此之外，对于用户的其他动作，仍然采用事件驱动方式工作。详情可参见 1.12 节、3.3 节和 11.1 节。

1.6 Windows 应用程序框架

Managed DirectX 和 XNA 程序以.NET Framework Windows 应用程序框架为基础，因此有必要复习一下 Windows 应用程序的基本框架。Windows 应用程序的执行总是从 Main()方法开始，主函数 Main()必须在一个类中。Windows 应用程序使用图形界面，一般都有一个窗体(Form)，它采用事件驱动方式工作。本节介绍 Windows 应用程序的基本框架。

【例 1.1】最简单的.NET Framework Windows 应用程序如下：

```
using System;                                          //引入命名空间
using System.Windows.Forms;
```

```
public class Form1:Form                                    //类定义
{    static void Main()                                    //主函数
   {Application.Run(new Form1());              }
}
```

类 Form1 以 Form 类为基类。Form 类是.NET Framework 中定义的窗体类，Form 类对象具有 Windows 应用程序窗口最基本的功能，有标题栏、系统菜单、最大化按钮、最小化按钮、关闭按钮和用户区。Form 类对象还是一个容器，在 Form 窗体中可放置其他控件，例如菜单控件和工具条控件等。System.Application 类的静态方法 Run 负责完成一个应用程序的初始化、运行和终止等功能，其参数是本程序使用的窗体 Form1 类对象，Run 方法还负责从操作系统接受事件，并把事件送到窗体中响应。窗体关闭，Run 方法退出，Windows 应用程序终止运行。

使用 1.7 节介绍的方法用 VS 2005 建立 Windows 应用程序项目，右击解决方案资源管理器窗口中的 Form1.cs 文件，用弹出的快捷菜单中的“删除”菜单项来删除 Form1.cs 文件。用例 1.1 的代码替换 Program.cs 中的源代码，编译运行后将看到一个空白窗体。

可以在 Form1 类中定义新的变量。由于主窗体关闭，程序也就结束了，因此定义在主窗体 Form1 类中的变量的生命周期和程序的生命周期是相同的。从这个意义上说，这些变量是全局变量。可以为 Form1 类定义构造函数，在构造函数中做一些初始化的工作，例如修改 Form1 标题栏中的标题。还可以在 Form1 中定义控件类的对象，这些控件将在 Form1 的用户区显示出来，因此在 Form1 中定义控件类的对象，也称作把控件放到窗体中。

【例 1.2】本例通过在窗体中增加了一个按钮(Button)控件，并为单击按钮事件增加事件处理函数，说明了如何在窗体中增加控件，如何修改控件属性，如何增加控件的事件处理函数。具体代码如下：

```
using System;
using System.Windows.Forms;
using System.Drawing;
public class Form1:Form
{   Button button1;          //生成 Button 类引用变量,和应用程序有相同的生命周期
    public Form1()           //构造函数
    {    //下句修改主窗体标题,不指明属性(方法)所属对象,默认为 Form1 类的属性(方法)
        Text = "我的第一个程序";                    //也可写为：this.Text = "我的第一个程序";
        button1 = new Button();                    //生成 Button 类对象
        button1.Location = new Point(25,25);       //修改 button1 属性 location,即按钮位置
        button1.Text = "确定";                     //修改 button1 属性 Text,即按钮标题
        //下句指定 button1_Click 函数是按钮单击事件的单击事件处理函数
        button1.Click += new System.EventHandler(button1_Click);
        this.Controls.Add(button1);                //按钮增加到窗体中,将在主窗体用户区显示出来
    }
    static void Main()
```

```
    {Application.Run(new Form1());
    }     //下面的函数是单击按钮事件处理函数,产生 Click 事件,将自动调用这个函数
    private void button1_Click(object sender, EventArgs e)
    {this.button1.Text = "单击了我";            }    //单击按钮后执行的语句
}
```

注意在窗体中增加控件类对象的步骤。第 1 步定义 Button 类变量 button1，这是 Form1 类的一个字段，由于主窗体关闭后程序也就结束，因此变量 button1 的生命周期和程序的生命周期是相同的，从这个意义上说，button1 是全局变量。第 2 步在构造函数中用 new 生成 Button 类对象。第 3 步在构造函数中修改 button1 的属性。第 4 步增加 button1 的事件处理函数，语句 button1.Click+=new System.EventHandler(button1_Click)指定按钮 Button1 事件 Click 事件处理函数是程序员编写的 button1_Click()函数，程序员应在事件处理函数 button1_Click()中增加具体的事件处理语句。这些步骤对于增加任何控件都是相同的。

1.7　可视化程序设计

使用 VS 2005 可以采用可视化方法设计 Windows 应用程序，可通过鼠标的拖动操作将 VS 2005 工具箱窗口中的控件放到窗体中，然后使用属性窗口或在程序中用语句修改控件属性，从而设计应用程序界面，为控件增加事件处理函数，最后完成指定的功能。使用 Managed DirectX 框架设计 3D 程序时，也用到了可视化方法，因此这里简单介绍一下这方面的内容，以方便读者理解。

【例 1.3】本例使用微软 VS 2005 用可视化方法创建 Windows 应用程序，在窗口中显示一行文字，增加两个按钮，单击标题为“红色”的按钮，显示的文本颜色变为红色，单击标题为“黑色”的按钮，显示的文本颜色变为黑色。实现步骤如下。

(1) 运行 VS 2005 程序，选择菜单“文件(F)|新建项目(P)…”菜单项，打开“新建项目”对话框。在“模板(T)”列表框中选择“Windows 应用程序”，单击“确定”按钮，创建项目。随后将出现图 1.1 所示的界面，其中有一个空白窗体(Form1)。
选择“文件(F)|全部保存(L)”菜单项，保存解决方案的所有文件。在“保存项目”对话框(见图 1.2)，如图选择名称和位置。然后单击“保存(S)”按钮，保存所有文件。最终的 VS 2005 界面如图 1.1 所示(已打开解决方案资源管理器窗口)。

(2) 若不在标题为“Forms.cs[设计]”的窗口，单击图 1.1 中标题为“Forms.cs[设计]”的窗口标签，返回标题为“Forms.cs[设计]”的窗口。向 Form1 窗体中添加控件需要使用工具箱窗口，若看不到，可以用“视图”|“工具箱”菜单项打开这个窗口(见图 1.3)。
双击工具箱窗口中“所有 Windows 窗体”标签下的 Label 控件，在窗体 Form1 中放置一个 Label 控件。该控件用来显示一行文本。可以用鼠标拖动控件 Label 到窗体的任意位置，并可拖动 Label 控点(控件边界上的小方框)改变控件的尺寸。

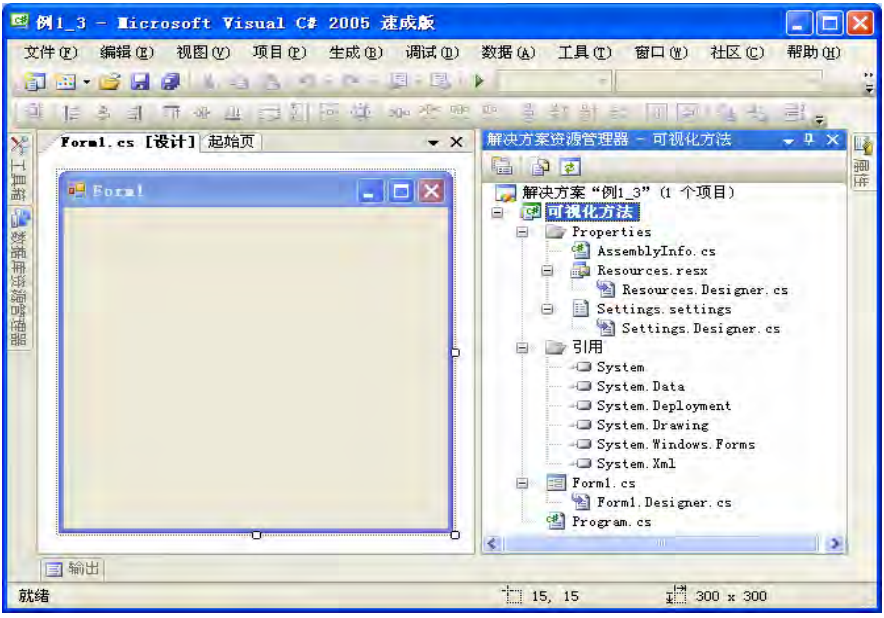


图 1.1　VS 2005 集成环境界面

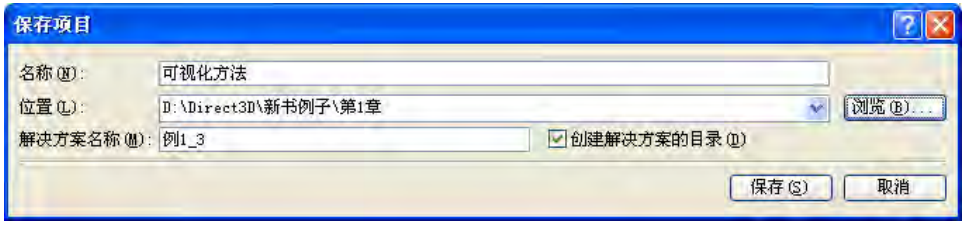


图 1.2　"保存项目"对话框

(3) 选择"视图"|"属性窗口"菜单项打开属性窗口(图 1.4)，属性窗口显示控件属性，其中左侧为属性名称，右侧为属性值。选中 Label 控件，在属性窗口中找到名称为 Text 的属性，在其右侧把 Text 属性值由"Label1"修改为"我的第一个程序"，在属性窗口选中 Font 属性，单击 Font 属性右侧的浏览按钮(…)，打开"字体"对话框，在对话框中可以修改 Label 控件显示字符的字体名称和字号等。也可以单击 Font 属性左边的+号，显示子属性。可编辑这些子属性。编辑完成后，单击 Font 属性左边的-号，隐藏 Font 的子属性。修改 ForeColor 属性可以修改 Label 控件显示字符的颜色。这是在设计阶段修改属性。

(4) 接下来在窗体中增加 3 个按钮，并为按钮增加单击事件函数。打开标题为"Forms.cs[设计]"的窗口，双击工具箱窗口中"所有 Windows 窗体"标签下的 Button 控件，Button 控件将放到 Form1 窗体中，可以拖动 Button 控件到指定位置。在 Form1 窗体中选中按

钮控件，用属性窗口修改 Button 的 Text 属性值分别为红色、黑色、退出。单击属性窗体上的第 4 个图标，打开事件窗口(见图 1.5)，显示 Button 控件所能响应的所有事件，其中左侧为事件名称，右侧为事件处理函数名称，如果右侧为空白，表示还没有指定事件处理函数，选中 Click 事件，双击右侧空白处，为 3 个按钮增加单击事件处理函数。

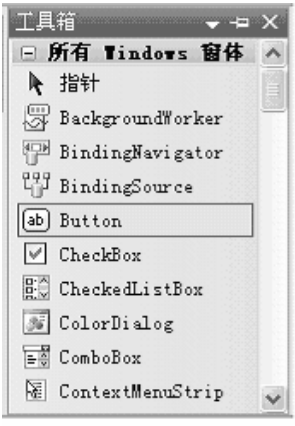


图 1.3　工具箱

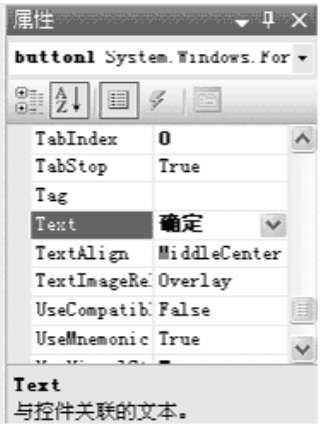


图 1.4　属性窗口

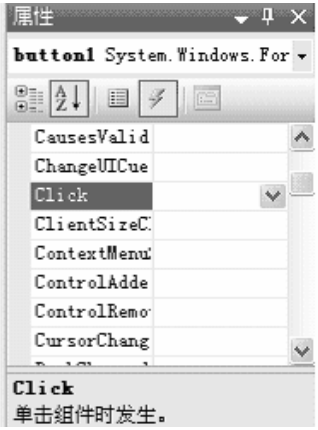


图 1.5　事件窗口

(5) 右击标题为“Forms.cs[设计]”窗口中的 Form1 窗体，在快捷菜单中选择菜单项“查看代码(C)”，便可打开 Form1.cs 文件。为 3 个按钮单击事件处理函数增加事件处理语句，如下所示：

```
private void button1_Click(object sender, EventArgs e)  //红色按钮事件处理函数
{   label1.ForeColor = Color.Red;//此行语句由程序员增加,方法的其余代码是自动增加的
}        //注意,label1 是控件的名字(label 的 Name 属性),用于区分不同的控件
private void button2_Click(object sender, EventArgs e)  //黑色按钮事件处理函数
{   label1.ForeColor = Color.Black;}  //运行阶段修改属性,Black 为 Color 结构的静态属性
private void button3_Click(object sender, EventArgs e)  //退出按钮事件处理函数
{   Close();      }                                      //调用窗体 Form1 类的方法 Close()
```

(6) 编译运行后，单击标题为“红色”的按钮，窗体用户区中文字符颜色变为红色。单击标题为“黑色”的按钮，中文字符颜色变为黑色。单击标题为“退出”的按钮，结束程序。

控件放到窗体后，VS 2005 集成环境在 Form1.Designer.cs 文件和 Form1.cs 文件中自动增加了一些语句。仔细查看这些增加的语句后，可以看出用可视化方法在窗体中增加 Button 按钮的步骤和上节所述步骤是一样的。这些步骤适用于增加任何控件。