R G D C 游 戏 开 发 课 程 体 系

第九艺术学院——游戏开发系列
COLLEGE OF THE NINTH ART

游戏软件开发基础

姚 磊 陈帼鸾 陈 洪 编著

- 中国电影电视技术学会数字特效与三维动画专业委员会
- 中国系统仿真学会数字娱乐专业委员会
- 中国文化创意产业技术创新联盟

推 荐 教 材

清华大学出版社

第九艺术学院——游戏开发系列

游戏软件开发基础

姚　磊　陈帼鸾　陈　洪　编　著

清华大学出版社
北　京

内 容 简 介

本书作为游戏软件开发基础图书，共分为 5 个部分：第一部分讲解 Windows 基础，帮助读者掌握 Windows 基本概念和程序框架、Windows 消息机制，使读者对整个 Windows 程序有个大体的认识。第二部分讲解 Win32 基本 GDI 绘图及 DirectDraw 基础，使读者初步掌握游戏图像的绘制。第三部分讲解 DirectInput 和 DirectX Audio，讲述如何在游戏程序中使用外设控制，包括键盘鼠标的操控等，以及如何在游戏中添加优美的音乐和逼真的音效。第四部分结合前面所学的知识，学习 RPG 游戏制作的技巧，以及 RPG 游戏中所涉及的各种游戏元素的实现原理。第五部分讲解了 ACT 游戏的制作，包括射击类游戏以及横版过关类的游戏，让读者把前面所学的知识做一个综合的运用，制作两款简单的动作游戏。

本书共 8 章内容，讲解过程深入浅出，具体全面，为读者将来深入学习游戏编程打下了坚实的基础。希望能够通过本书给广大的 Windows 游戏开发爱好者提供一个全面了解 Windows 游戏开发的机会，帮助他们在自己喜爱的事业上找到努力的方向。更希望为中国游戏事业的发展提供更多的人才。

图书在版编目(CIP)数据

游戏软件开发基础/姚磊，陈帼鸾，陈洪编著. —北京：清华大学出版社，2010.1
(第九艺术学院——游戏开发系列)
ISBN 978-7-302-21805-0

Ⅰ. 游…　Ⅱ. ①姚…　②陈…　③陈…　Ⅲ. 游戏—软件开发—高等学校—教材　Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字(2009)第 238768 号

责任编辑：张彦青　张丽娜
装帧设计：杨玉兰
责任印制：杨　艳
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：清华大学印刷厂
装 订 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：185×230　**印　张**：19　**字　数**：405 千字
版　　次：2010 年 1 月第 1 版　　**印　　次**：2010 年 1 月第1 次印刷
印　　数：1～4000
定　　价：38.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　　产品编号：035310-01

前　言

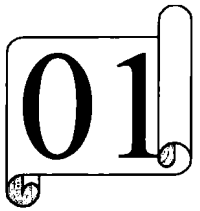
当今游戏产业发展日益加快，游戏开发技术也在不断地跳跃式发展，而且已经到了一个技术日趋活跃的重要时刻，这是作为游戏程序员的深切感受。如今每几个月就要推出一种新型的 CPU、显卡和其他的硬件设备，将技术不断地向前推进。随之而来的就是玩家期待游戏开发者制作的游戏能够使用新技术，新技术将减少未来视频游戏发展的障碍，未来，唯一的限制因素将是我们的知识和想象力。我们编写本书的目的，正是为了向读者提供游戏开发的基础知识，把读者带入游戏编程制作领域，让读者能从本书中得到鼓舞、指导和启迪。

很少有软件工程领域对硬件、软件和程序员本身的要求像游戏编程一样多。将数学、物理、人工智能、图像、声音、音乐、GUI 和数据结构等内容完美地结合在一起是非常困难和复杂的工作，这也正是游戏编程的难点所在。现今我们拥有了制作逼真游戏的技术，但这些技术并不容易掌握，需要游戏开发者对多方面的知识进行刻苦学习，其中学习 Windows 游戏程序设计尤为重要。我们编写本书的目的也是源于这点，就是要求作者以 Windows 程序原理为起点，扩展开来，讲解如何编写常见的 Windows 二维游戏程序，让读者从图像、声音、控制等多个角度学习二维游戏程序的设计，带领读者实现几款经典的二维游戏。

本书旨在让初学者逐步掌握 Windows 二维游戏程序设计的技巧与要领，适合游戏行业的相关初学者学习游戏编程方面的专业知识，也适合高等院校和各种培训机构作为培训游戏人才的教材。

学习游戏程序设计之路任重而道远，本书的任务只是把学习者领进门，而日后的漫长修行之路还是要靠读者个人的不断努力。好了，相信读者已经准备就绪，那就让我们开始游戏编程之旅吧！

目　录



Windows 基础

教学目标

- 了解 Windows 基本概念
- 掌握 Windows 基本框架
- 掌握 Windows 消息机制

1.1 Windows 概述

现在，Windows 已众所周知，似乎不需要再介绍，然而人们很容易忘记 Windows 给办公室和家庭带来的重大改变。Windows 在其早期的发展历程中曾经走过一段坎坷的道路，征服台式计算机市场的前途曾一度相当渺茫。下面介绍 Windows 的发展历程。

1. Windows 1.0

Windows 的发展始于 Windows 1.0 版本。这是 Microsoft 在商业视窗操作系统的第一次尝试，当然这也是一个非常失败的产品。Windows 1.0 完全建立在 DOS 基础上(这就是一个错误)，不能执行多任务，运行速度很慢，外观看上去也很差劲。其外观实际上可能是失败的最重要原因。此外，失败的原因还在于 Windows 1.0 与那个时代的 80286 计算机(或更差的 8086)所能提供的性能相比，实在需要更高的硬件、图像和声音性能。

2. Windows 2.0

随着 Microsoft 稳步前进，很快就推出了 Windows 2.0。运行 Windows 2.0 测试演示版可以发现，它装载了多个应用程序，看上去似乎还能够工作。但是，那时 IBM 推出了 PM。

PM 看上去要好得多，它是建立在比 Windows 2.0 先进得多的操作系统 OS/2 基础上的，而 Windows 2.0 依然是建立在 DOS 基础上的视窗管理器。

3. Windows 3.x

1990 年，视窗操作系统终于发生了翻天覆地的变化，因为 Windows 3.0 出世了，而且其表现确实非常出色。尽管它仍然赶不上 Mac OS 的标准，但是谁还在意呢？(真正的程序员都憎恨 Mac)。有了 Windows 3.0，软件开发人员终于可以在 PC 机上创建迷人的应用程序了，而商用应用程序也开始脱离 DOS。这成了 PC 机的转折点，终于将 Mac 完全排除在商用程序之外了，而后也将其挤出台式机出版业(那时，Apple 公司每 5 分钟就推出一种新硬件)。

尽管 Windows 3.0 工作性能良好，却还是存在许多的问题和软件漏洞，但从技术上说它已是 Windows 2.0 之后的巨大突破。为了解决这些问题和软件漏洞，Microsoft 推出了 Windows 3.1，开始公关部和市场部打算称为 Windows 4.0，但是，Microsoft 决定只简单地称为 Windows 3.1，因为它不足以称为升级的换代版本。它还没有做到市场部广告宣传的那样棒。

Windows 3.1 非常可靠。它带有多媒体扩展以提供音频和视频支持，而且它还是一个出色的、全面的操作系统，用户能够以统一的方式来工作。另外，Windows 3.x 还存在一些其他版本，如可以支持网络的 Windows 3.11(适用于工作的 Windows)。这时唯一的问题在于 Windows 3.1 仍然是一个 DOS 应用程序，运行在 DOS 扩展器上。

4. Windows 95

1995 年，当游戏编程行业还在唱“DOS 永存!”的赞歌时，Windows 95 终于推出了。它是一个真正 32 位、多任务、多线程的操作系统。虽然 Windows 95 中还残存一些 16 位代码，但是在很大程度上，Windows 95 已经成为 PC 机的终极开发和发布平台。

当然，Windows NT 3.0 也同时推出，但那时 NT 对于大多数用户来讲还是不可用的，因此这里也就不再赘述。

几乎一夜之间，Windows 95 就改变了整个计算机行业。的确，目前还有一些公司仍然在使用 Windows 3.1，但是 Windows 95 使得基于 Intel 的 PC 机成为除游戏程序之外的所有应用程序的选择。尽管游戏程序员知道 DOS 退出游戏编程行业只是个时间的问题了，但是 DOS 还是他们进行游戏编程的核心。

1996 年，Microsoft 发布了 Game SDK(游戏软件开发工具包)，这基本上就是 DirectX 的第一个版本。这种技术仅能在 Windows 95 环境下工作，但是它运行起来的确太慢了，甚

至竞争不过 DOS 游戏(如 Doom 和 Duke Nukem 等)。游戏开发人员继续使用 DOS 来开发游戏，但是他们知道 DirectX 具有足够快的运行速度，从而使游戏能够流畅地运行在 PC 机上已为时不远。

到了 3.0 版本，DirectX 的运行速度在同样的计算机上已经和 DOS32 一样快了。到了 5.0 版本，DirectX 已经相当完美，实现了该技术最初的承诺。现在要意识到：Win32/DirectX 是 PC 机上开发游戏的唯一方式，这是历史的选择。

5. Windows 98

1998 年中期，Windows 98 推出了。这至多是一个改进的版本，而不像 Windows 95 那样是一个换代的产品，但毫无疑问它也占有很重要的地位。Windows 98 像一辆旧车改装的高速汽车——实际上它是一头皮毛圆润光滑、脚力持久、飞奔跳动的驴。它是全 32 位的，能够支持想做的所有事情，并具有无限扩充能力。它很好地集成了 DirectX、3D 图形、网络以及 Internet。

Windows 98 和 Windows 95 相比也非常稳定。当然 Windows 98 仍然经常死机，但可以相信的是，死机次数比 Windows 95 少了许多。Windows 98 对即插即用支持得很好，并且能够很好地运行。

6. Windows NT

现在来讨论一下 Windows NT。Windows NT 要比 Windows 9X 严谨得多，这使 Windows 9X 逐渐退出历史舞台。Windows NT 5.0 最酷的是它完全支持即插即用和 Win32/DirectX，因此使用 DirectX 为 Windows 9X 编写的应用程序可以在 Windows NT 5.0 或更高版本上运行。这的确是个好消息，因为这使得编写 PC 游戏的开发人员现在具有最大的市场潜力。

也就是说，如果使用 DirectX(或其他工具)编写了一个 Win32 应用程序，它完全可以在 Windows 95、Windows 98 和 Windows NT 5.0 或更高版本上运行。这可是件好事情，因为这样在早期所学到的任何东西至少可以应用到三种操作系统上，也可以运行于安装 Windows NT 和 DirectX 的其他计算机上，如 DEC 的 Alphas，还有 Windows CE——DirectX 和 Win32 衍生的运行于其他系统上的操作系统。

7. Windows 2000

Windows 2000 平台操作系统采用 NT 技术，并在其上做了大量的改进，使 Windows 2000 操作系统平台比此前的 Windows 操作系统平台更加可靠，更易扩展，更易部署，更易管理和更易使用。值得一提的是，Windows 2000 还提供了对多国语言的支持。

Windows 2000 还提供了增强的网络管理，利用“网上邻居”窗口左侧的“添加网络组件”命令，可以为网络增加网络管理和监视工具。该工具的作用是监视网络设备的活动，并且向网络控制台工作站汇报情况。

Windows 2000 家族有两大类平台(共四种操作系统)：第一类是工作站平台 Windows 2000 Professional；第二类是服务器平台。Windows 2000 家族的服务器平台有如下三种。

- Windows 2000 Server：除了包含 Windows 2000 Professional 的所有特性外，它还提供简单的网络管理服务。
- Windows 2000 Advanced Server：除了包含 Windows 2000 Server 的所有特性外，它还提供更好的可扩展性和有效性，并支持更多的内存、处理器以及群集。
- Windows 2000 Data Center Server：除了包含所有 Windows 2000 Advanced Server 的特性外，它还提供更多的内存和处理器的支持，适用于大型数据仓库和在线事务处理等重要应用。

8. Windows XP

Windows XP 于 2001 年 10 月 25 日正式发布，XP 的意思是 eXPerience，即体验。与 Windows 2000 相同，Windows XP 基于 NT 技术，是纯 32 位的操作系统，其功能更健壮、更稳定。

Windows XP 在网络特性上的增强有不少值得称道的方面。不少用户都知道 Windows 2000 中的脱机文件功能，Windows XP 对其做了进一步优化，并可将脱机文件加密，使其更为安全。在兼容性方面，不少用户都知道 Windows 2000 的一个弊病就是与不少应用软件(特别是游戏软件)不兼容。但是，在 Windows XP 中不存在这个问题，在其他 Windows 系统中可以运行的软件几乎都可以在 Windows XP 中运行。此外，Windows XP 的核心代码基于 Windows 2000，所以从 Windows NT 4.0/2000 上进行升级安装是十分容易的。Windows XP 提供了 4 个适用于不同用途的版本：个人版(支持 1 个 CPU)、专业版(支持 2 个 CPU)、服务器版(支持 4 个 CPU)和高级服务器版(支持 8 个 CPU)。

9. Windows Vista

Windows Vista 是继 Windows XP 之后 Microsoft 推出的新一代操作系统，以前叫做 Longhorn(微软当初内部的代号)，如图 1-1 所示。Microsoft 对外宣布正式名称是 Windows Vista。作为微软的最新操作系统，Windows Vista 第一次在操作系统中引入了“Life Immersion”概念，即在系统中集成许多人性的因素，一切以人为本，使得操作系统尽最大可能贴近用户，了解用户的感受，从而方便用户使用。

在 Windows Vista 操作系统上，可以利用 DirectX 10 中的 XNA 开发 Xbox 360 上的单

机游戏，可见 Windows Vista 操作系统为以后更方便地开发各种游戏提供了一个很好的平台。

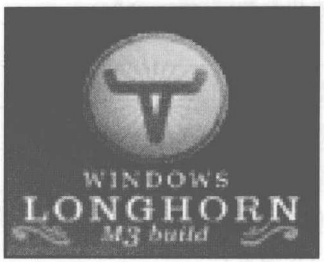


图 1-1　Longhorn 标志

1.2　Windows 的命名规则及窗口的创建

在正式学习 Windows 编程之前，肯定需要熟悉一下 Windows 的命名规则，这样对学习整个 Windows 会有事半功倍的效果。

1.2.1　匈牙利规则

如果正在运作一个像 Microsoft 一样的公司，大约有几千个程序员都在干不同的项目，那么就应当提出一个编写代码的标准方式。否则，结果将是一片混乱。因此一个名叫 Charles Simonyi 的人被委托创立了一套编写 Microsoft 代码的规范。这个规范已经被用作为编写代码的基本指导说明书。所有 Microsoft 的 API、界面、技术文件等都采用这个规范。

这个规范通常被称为匈牙利符号表示法，因为 Charles Simonyi 是匈牙利人。读者必须了解这个规范，以便于能够阅读 Microsoft 代码。匈牙利符号表示法包括许多与变量、函数、类型和常量、类、参数命名有关的约定。

表 1-1 给出了匈牙利符号表示法使用的前缀代码。这些代码在大多数情况下一般用于前缀变量名，其他约定根据名称确定，相应解释可以参考本表。

表 1-1　匈牙利符号表示法的前缀代码

前　缀	数据类型(基本类型)
c	字符
by	字节(无符号字符)

续表

前　缀	数据类型(基本类型)
n	短整数和整数(表示一个数)
i	整数
x，y	短整数(通常用于表示 x 坐标和 y 坐标)
cx，cy	短整数(通常用于表示 x 和 y 的长度，c 表示计数)
b	布尔型(整数)
w	UINT(无符号整数)和 WORD(无符号字)
l	LONG(长整数)
dw	DWORD(无符号长整数)
fn	函数指针
s	串
sz，str	以 0 字节终止的字符串
lp	32 位的长整数指针
h	编号(常用于表示 Windows 对象)
msg	消息

1. 变量的命名

应用匈牙利符号表示法，变量可用表 1-1 中的前缀代码来表示。另外，当一个变量是由一个或几个子名构成时，每一个子名都要以大写字母开头。下面是几个例子：

```
char  *szFileName;  //以 0 字节终止的字符串
int   *lpiDate;     //指向 32 位类型的 int 指针
BOOL  bSemaphore;   //布尔变量
WORD  dwMaxCount;   // 32 位的无符号整数
```

全局变量一般用 g 作开头，举例如下：

```
int g_iXPos;      //一个全局 x 坐标的变量
int g_iTimer;     //一个全局的时间变量
char *g_szString;// 一个全局的以 0 字节终止的字符串
```

全局变量以 g_开头，或者有时就只用 g。

2. 函数的命名

函数和变量命名方式相同，但是没有前缀，换句话说，子名的第一个字母要大写。下面是两个例子：

```
int PlotPixel(int ix,int iy,int ic);
void *MemScan(char *szString);
```

而且，下划线是非法的，例如，下面的函数名表示是无效的匈牙利符号表示法：

```
int Get_Pixel(int ix,int iy);
```

3. 类型和常量的命名

所有的类型和常量都是大写字母，但名字中允许使用下划线。例如：

```
const LONG NUM_SECTORS = 100;
#define MAX_CELLS 64;
#define POWERUNIT 100;
typedef unsigned char UCHAR;//用户自定义类型
```

尽管大多数 Microsoft 程序员不使用下划线，但大部分程序员还是喜欢用，因为这样能使名字更具有可读性。在 C++中，关键字 const 不只一个意思。在前面的代码行中，它用来创建一个常数变量。这和#define 相似，但是它增加了类型信息这个特性，const 不仅仅像#define 一样是一个简单的预处理文本替换，它更像是一个变量，它允许编译器进行类型检查和替换。

4. 类的命名

类命名的约定可能要麻烦一些。但也有很多人在使用这个约定，并独立地进行补充。不管怎样，所有 C++的类必须以大写字母 C 为前缀，类名字的每一个子名的第一个字母都必须大写。示例如下：

```
class CVector
{
public:
CVector(){ix=iy=yz=imagnitude = 0;}
CVector(int x,int y,int z)
{
ix = x;
iy = y;
iz = z;
}
private:
int ix,iy,iz;//向量的三个分量
int imagnitude;//向量
};
```

5. 参数的命名

函数的参数命名和标准变量命名的约定相同，但也不总是如此。例如下面的例子给出了一个函数定义：

```
UCHAR GetPixel(int x,int y);
```

这种情况下，更准确的匈牙利函数原型是：

```
UCHAR GetPixel(int ix,int iy);
```

但作者认为这并没有什么两样。

最后，甚至可能都看不到这些变量名，而仅仅能够看到类型，如下所示：

```
UCHAR GetPixel(int, int);
```

当然，这仅仅是原型使用的形式，真正的函数声明必须带有可赋值的变量名。

1.2.2 Windows 窗口的创建

创建窗口的示例代码如下：

```
#include <windows.h>
long FAR PASCAL WindowProc(HWND hWnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
    switch(message)
    {

    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;
    case WM_KEYDOWN:
        switch(wParam)
        {
        case VK_ESCAPE:
            PostMessage(hWnd, WM_QUIT, 0, 0);
            break;
        }
        break;
    }
    return DefWindowProc(hWnd, message, wParam,lParam);
}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
```

```
                    LPSTR lpCmdLine, int nCmdShow)
{
    MSG msg;          //定义一个消息的对象
    HWND hWnd;        //窗口句柄
    WNDCLASS wc;      //定义窗口类
    HDC hdc;
    //创建窗口类
    wc.style=CS_HREDRAW | CS_VREDRAW;   //支持水平和垂直重绘
    wc.lpfnWndProc=WindowProc;          //相应的消息处理函数
    wc.cbClsExtra=0;                    //类附加内存空间
    wc.cbWndExtra=0;                    //窗口附加内存空间
    wc.hInstance=hInstance;             //窗口的事例句柄
    wc.hIcon=LoadIcon(hInstance,MAKEINTRESOURCE(IDI_ICON1));//窗口的小图标
    wc.hCursor=NULL;                                        //窗口的鼠标形状
    wc.hbrBackground=(HBRUSH)GetStockObject(BLACK_BRUSH);//背景颜色
    wc.lpszMenuName=NULL;                                   //窗口菜单
    wc.lpszClassName="CreateWindows";                       //窗口名称
    RegisterClass(&wc);                                     //注册窗口句柄
    hWnd=CreateWindowEx(WS_EX_TOPMOST,//窗口总显示在顶部
        "CreateWindows", //窗口的类名
        "!!pomm!!",       //窗口的标题
        WS_POPUP,         //窗口的风格
        0,                //X 轴初始化设定坐标
        0,                //Y 轴初始化设定坐标
        1024,             //宽度初始化
        768,              //高度初始化
        NULL,             //父窗口句柄
        NULL,             //窗口菜单句柄
        hInstance,        //实例句柄
        NULL);            //附加信息
    if(!hWnd)//判断窗口是否建立成功
    {
        return FALSE;
    }
    ShowWindow(hWnd, nCmdShow);//显示窗口
    UpdateWindow(hWnd);//更新窗口

    while(true)
    {
        //接受系统消息(&msg 为 MSG 类型的信息结构体,NULL 为窗口句柄,
        //0,0 表示接受所有的窗口信息)
        if(PeekMessage(&msg, NULL, 0U, 0U, PM_REMOVE))
        {
            if(msg.message==WM_QUIT)
                break;
```

```
            TranslateMessage(&msg);  //转换虚拟键码
            DispatchMessage(&msg);   //转到消息处理函数
        }
        else
        {
           WaitMessage();
        }
    }
    return 1;
}
```

下面就对上面的代码进行详细的分析：

所有的 Windows 程序都以 WinMain()开始，这和简单直观的 DOS 程序都以 Main()开始一样。WinMain()中的内容取决于开发者。如果开发者愿意的话，可以创建一个窗口、开始处理事件并在屏幕上画一些东西。另外，还可以调用几百个(或者是几千个)Win32 API 函数中的一个。创建 Windows 程序有两种方式：使用 Microsoft 基础类(Microsoft Foundation Classes，MFC)；或者使用软件开发工具包(Software Development Kit，SDK)。MFC 完全基于 C++类，要比以前的游戏编程所需的工具复杂得多，功能和难度也要强大和复杂很多倍。而 SDK 是一个可管理程序包，可以在一到两周内学会(至少初步学会)，并且使用了简单明了的 C 语言。因此，在本书中使用的工具是 SDK。

第一个引用<windows.h>实际上包括所有的 Windows 头文件。Windows 有许多这样的头文件，这有点像包含宏，可以节省许多手工显式包含头文件的时间。

下面就介绍最重要的部分——所有 Windows 应用程序的主要入口位置 WinMain()：

```
int WINAPI WinMain( HINSTANCE  hInstance,
  HINSTANCE hPrevInstance,
  LPSTR     lpCmdLine,
  int       nCmdShow);
```

首先，应当注意到奇怪的 WINAPI 声明符，这相当于 PASCAL 函数声明符，它强制参数从左边向右边传递，而不是像默认的 CDECL 声明符那样参数从右到左转移。但是，PASCAL 调用约定声明已经过时了，WINAPI 代替了该函数。必须使用 WinMain()的 WINAPI 声明符，否则，将向函数返回一个不正确的参数并终止开始程序。

下面详细介绍一下每个参数。

- hInstance：该参数是 Windows 为应用程序生成的一个实例句柄。实例是一个用来跟踪资源的指针或数。本例中，hInstance 就像一个名字或地址一样，用来跟踪应用程序。
- hPrevInstance：该参数已经不再使用了，但是在 Windows 的旧版本中，它跟踪