动漫游戏设计
系列教程

Direct3D
实时渲染技术

曾凡喜　周　炜　潘运亮　等编著

中国水利水电出版社
www.waterpub.com.cn

动漫游戏设计
系列教程

Direct3D
实时渲染技术

曾凡喜　周　炜　潘运亮　等编著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书不仅系统地讲述了 DirectX 接口和函数的使用，而且简单剖析了其背后隐含的 3D 数学和图形学原理，并对 DirectX 接口介绍得十分详细，基础部分的内容不亚于帮助文档。

注重实践，实例丰富。部分重要的程序代码在书中列出，既突出了代码的重要性，又没有缩减整本书的文字量。读者可以对照程序代码看书，有利于加深对 Direct3D 的理解和掌握。

本书非常适合初学者或有一定基础的读者学习 Direct3D 实时渲染技术时使用。

本书提供实例的完整源代码，读者可以从中国水利水电出版社网站或万水书苑上免费下载，网址为：http://www.waterpub.com.cn/softdown/和 http://www.wsbookshow.com。

图书在版编目（CIP）数据

Direct3D 实时渲染技术 / 曾凡喜等编著. —北京：中国水利水电出版社，2009

（动漫游戏设计系列教程）

ISBN 978-7-5084-6419-0

Ⅰ. D… Ⅱ. 曾… Ⅲ. 多媒体—软件工具，Direct3D Ⅳ. TP311.56

中国版本图书馆 CIP 数据核字（2009）第 049418 号

策划编辑：石永峰　责任编辑：张玉玲　封面设计：无极书装

书　　名	动漫游戏设计系列教程 Direct3D 实时渲染技术
作　　者	曾凡喜　周　炜　潘运亮　等编著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	210mm×285mm　16 开本　18.75 印张　491 千字
版　　次	2009 年 5 月第 1 版　2009 年 5 月第 1 次印刷
印　　数	0001—3000 册
定　　价	35.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

丛书序

互联网的快速普及为游戏的发展奠定了良好的基础，游戏已经成为互联网发展的重要应用之一。现在，游戏产业正以不容置疑的速度发展，已经成为 IT 领域中的一个重要产业。不过，和 IT 领域中其他产业的发展类似，我国游戏产业的迅猛发展也同样遭遇了人才瓶颈。目前不论是金山等游戏开发厂商，还是靠游戏而成名的盛大，或是靠游戏而风光于美国纳斯达克的网易，都已在全国广觅游戏人才，游戏人才已经变得炙手可热。

在我国的 IT 领域，市场的发展也必然会对人才不断地提出新要求，前几年是硬件人才、网络人才，接下来是软件人才，而就目前的市场需求看，毫无疑问游戏人才是最炙手可热的市场需求点之一。据了解，我国当前游戏行业最为缺乏的是游戏设计与开发工程师，为了迅速摆脱这一发展困境，我们策划出版了这套系统介绍游戏开发技术及其应用的丛书，以满足市场的广泛需求。

本丛书围绕游戏开发整个技术体系的各个方面展开，系统地介绍了三维游戏引擎设计技术及其应用、三维图形实时渲染技术及其应用、游戏中的人工智能技术及其应用、游戏动画和音效编程技术及其应用，以及游戏中的网络编程技术及其应用等，形成了一个完整的最新的游戏开发技术体系结构，有助于读者全面地学习游戏开发中最新最流行的技术和理论。当然如果读者只想了解游戏开发中某一方面的技术，则可以单一学习丛书中的某一分册，因为丛书的各分册相对独立。同时本丛书在理论上和实践上都有比较鲜明的特点，因为本丛书的作者不仅在理论上有较深的造诣，阅读过大量游戏编程技术的硕博期刊论文和外文书籍，并在相关方面进行理论研究，而且还从事相关方面的项目和游戏及虚拟现实公司的具体产品的开发，对国外的开放源代码有过较为深入的分析。如果读者想学习到当今计算机图形学和人工智能方面的最新理论，为自己以后的理论研究作好积累，本丛书无疑可以起到抛砖引玉的作用。如果读者想在学习完本丛书之后进入实际项目的开发，抑或想去游戏公司应聘，本丛书也可以在职场上助你一臂之力。丛

丛书序

书的主要风格是在理论上深入，涉及游戏编程相关的各种高级技术和国外大学相关课程的理论内容，同时在实践中浅出，以某些具体的应用为实例，把理论付诸于实践，并在实例中讲述实际编程中的各种技巧。正因为理论和实践并重，本丛书的读者群比较广泛，可以是高校中从事计算机图形学和虚拟现实理论与算法研究的研究人员，也可以是想以后从事游戏编程工作的学生，也可以是已经从事游戏行业想进一步完善知识体系的开发人员。同时，本丛书也可以作为高校和培训学校的教材。

丛书编委会

前　言

记得我 2005 年初学习 Direct3D 时，国内没有一本介绍实时图形渲染技术的中文书籍，那时仅靠着阅读 DirectX SDK 帮助文档学习 DirectX 和 3D 图形相关知识，整个过程是有些困难的。诚然，要深入了解 Direct3D，不参考阅读 DirectX SDK 帮助文档是不行的。但对于初学者来说，不能不说这提高了学习者的门槛。因为它要求读者具备较强的英文阅读能力和较深的程序理解能力。如果读者不具备这方面的条件，难免会对这些帮助文档望而生畏。于是便产生了编写一本有关 DirectX 技术的中文书籍的想法。

游戏产业是近几年较为火爆的 IT 产业之一，也带动游戏编程成为了编程领域的热点，越来越多的人参与到这个行业中，也促进了相关中文书籍的诞生，但大多数书籍都不适合初学者。有的书是从国外翻译过来的，翻译过程中造成了语句不通顺、文字习惯不符合国内读者的阅读习惯等。有的书本讲得太难太深入，让初学者望而生畏。因此，编写一本符合初学者要求，内容较为浅显的书籍应该是非常必要且具有市场价值的。

笔者在编写本书的过程中，突出了以下特点：

（1）内容全面，适合入门。学习 Direct3D 编程不同于其他编程，要求具备的相关基础知识有计算机图形学、线性代数等。因此本书不仅系统地讲述了 DirectX 接口和函数的使用，还简单剖析了其背后隐含的 3D 数学和图形学原理。另外，本书对 DirectX 接口介绍十分详细，基础部分的内容不亚于帮助文档。因此，本书非常适合初学者学习。

（2）注重实践，实例丰富。部分重要的程序代码在书中列出，既突出了代码的重要性，又没有缩减整本书的文字量。读者可以对照程序代码看书，有利于加深对 Direct3D 的理解和掌握。

本书主要由曾凡喜编写，另外参加本书部分编写工作的还有周炜、潘运亮、林晓珊、黄卓、李鑫、王克杰、庄东填、张晋宝、林丽、王小青、赵应丁、郝思嘉、李俊峰、童剑等。由于作者水平有限，再加上时间仓促，书中疏漏甚至错误之处在所难免，恳请广大读者批评指正。我们的电子邮箱是 xinyuanxuan@263.net。

编 者

2009年1月

目 录

目 录

目 录

第 1 章
Windows 程序设计基础

主要内容：在讲述 DirectX 编程的知识之前，先简要地介绍 Windows 编程模式。因为本书所有章节的程序实例都是按 Win32 程序模式来编写的，所以本章将剖析一个完整的 Win32 程序。Windows 是一个多任务多线程操作系统，也是一个事件驱动系统。与 DOS 程序不同，Windows 程序等待用户执行某一操作，从而引发某一相应事件。

多任务的含义是多个应用程序可以同时执行，通常您不需要关心正在运行的其他应用程序，Windows 将会处理它们。您只需要关心自己的应用程序，并处理其中的消息或事件。在 Windows 3.0/3.1 系统中，并不完全这样，这些低级版本的 Windows 不是真正意义上的多任务操作系统，执行一个应用程序时不能执行其他的应用程序。在这些版本下运行的应用程序有些迟滞。如果其他应用程序过分占用了系统，您的应用程序将不能做任何事情。但是，Windows 9X 以上的系统中没有这样的问题。

本书的目标是在 Windows 平台上运行支持 DirectX 的图形渲染程序，无须知道太多的 Windows API 以及其他较深奥的 Windows 编程知识。

本章重点：

- Windows 应用程序框架
- 内存管理
- 类的概念
- 对象的构造和析构

1.1 一个完整的 Windows 应用程序

Windows 应用程序具有相对稳定的结构，也就是说 Windows 程序是有一定框架的，程序员要做的事情就是用特定的内容来填充这个框架。

所有 Windows 程序都是从 Winmain()函数开始的，这个函数与 DOS/UNIX 程序的主函数 main()的功能是一样的，它代表了程序的入口。其函数原型如下：

```
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,PSTR szCmdLine,
int iCmdShow)
```

这个函数看上去比 main()烦琐得多，如果是第一次看见这个函数肯定会感到一头雾水，但是没关系，我们来看看该函数中各个参数的意义。hInstance 是程序实例的句柄，句柄可以理解为该应用程序的标志，它跟指针类似但又不同于指针，它的意义只有操作系统知道，也就是说操作系统通过 hInstance 就可以找到某个应用程序；hPrevInstance 是前一个实例的句柄；szCmdLine 是命令行参数，在这里可以不予理会；iCmdShow 是窗口的显示方式。我们没有必要清楚每个参数的具体意义，在整个应用程序中用得最多的是 hInstance 参数，只要弄明白这个参数是应用程序的句柄即可。

下面给出 Win32 程序中 Winmain()函数的完整代码：

```
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPTSTR lpCmdLine,
                   int nCmdShow)
{
    // TODO: 在此放置代码
    MSG msg;
    HACCEL hAccelTable;
    // 初始化全局字符串
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadString(hInstance, IDC_ZFXDREAM, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);
    // 执行应用程序初始化
    if (!InitInstance(hInstance, nCmdShow))
    {
        return FALSE;
    }
    hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_ZFXDREAM);
    // 主消息循环
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
```

```
    }
    return (int) msg.wParam;
}
```

首先来看主消息循环，GetMessage()函数获取 Windows 系统发送过来的消息（比如你在应用程序窗口某处点击一下鼠标，或按下某键，这些操作都将以消息的形式通过 Windows 系统传递给应用程序）。主循环 while()将一直执行，直到 GetMessage()函数返回值为 0。GetMessage()函数是主事件循环的核心，它唯一的用途是从事件队列中获得下一条消息并在循环体中对其进行处理。GetMessage()函数有 4 个参数，我们只关心第一个参数，其他参数一般设置为 NULL 或 0。它的函数原型为：

```
BOOL GetMessage(LPMSG lpMsg,
    HWND hWnd,
    UINT wMsgFilterMin,
    UINT wMsgFilterMax
);
```

参数 msg 是一个结构体，它保存了与消息有关的所有内容，其结构原型为：

```
typedef struct {
    HWND hwnd;                  // 发生事件的窗口
    UINT message;               // 消息 ID（消息类型）
    WPARAM wParam;              // 更详细的消息信息，与消息类型有关
    LPARAM lParam;              // 更详细的消息信息，与消息类型有关
    DWORD time;                 // 消息发生的时间
    POINT pt;                   // 消息发生时的鼠标位置
} MSG, *PMSG;
```

再看 MyRegisterClass()函数，它的作用是注册一个窗口类，下面给出这个函数的全部代码：

```
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEX wcex;
    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.style = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc = (WNDPROC)WndProc;
    wcex.cbClsExtra = 0;
    wcex.cbWndExtra = 0;
    wcex.hInstance = hInstance;
    wcex.hIcon = LoadIcon(hInstance, (LPCTSTR)IDI_WIN32_INSTANCE);
    wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
    wcex.lpszMenuName = (LPCTSTR)IDC_WIN32_INSTANCE;
    wcex.lpszClassName = szWindowClass;
    wcex.hIconSm = LoadIcon(wcex.hInstance, (LPCTSTR)IDI_SMALL);

    return RegisterClassEx(&wcex);
}
```

WNDCLASSEX 是窗口类结构体，如下：

```
typedef struct {
    UINT cbSize;
    UINT style;
    WNDPROC lpfnWndProc;
    int cbClsExtra;
    int cbWndExtra;
    HINSTANCE hInstance;
    HICON hIcon;
    HCURSOR hCursor;
    HBRUSH hbrBackground;
    LPCTSTR lpszMenuName;
    LPCTSTR lpszClassName;
    HICON hIconSm;
} WNDCLASSEX, *PWNDCLASSEX;
```

其成员有：

- cbSize：WNDCLASSEX 结构体的大小，一般设为 sizeof(WNDCLASSEX)。
- style：用来定义窗口的行为。Win32 程序赋予的默认值为 CS_HREDRAW | CS_VREDRAW。
- lpfnWndProc：一个函数指针，指向与这个窗口类绑定在一起的处理窗口消息的函数。
- cbClsExtra 和 cbWndExtra：为窗口类和窗口对象分配多余的内存空间量的大小。很少使用到这两个参数，一般设为 0。
- hInstance：应用程序的实例句柄。可以使用 GetModuleHandle()来得到它，也可以从 Win32 程序的入口函数 WinMain 那里得到它。当然，也可以把它设为 NULL（不知道有什么用）。
- hIcon、hCursor、hbrBackground：设置默认的图标、鼠标、背景颜色句柄。不过在这里设置这些其实并不怎么重要，因为可以在后面定制自己的渲染方法。
- lpszMenuName：指向菜单资源标志符的字符串指针。
- lpszClassName：窗口类的名字。可以通过这个名字来创建以这个窗口类为模板的窗口，甚至可以通过这个名字来得到窗口的句柄。

设置好窗口类结构的内容后，使用 RegisterClass(const WNDCLASS *lpWndClass)函数来注册它。关闭窗口后可以用 UnregisterClass(LPCSTR lpClassName, HINSTANCE hInstance)来撤消注册。

注册完一个窗口类，下面的步骤就是调用应用程序初始化函数 InitInstance()，在这个过程中，需要完成的任务主要有创建和显示主窗口，其代码为：

```
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
   HWND hWnd;
   hInst = hInstance;              // 将实例句柄存储在全局变量中
   hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
      CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);
   if (!hWnd)
   {
      return FALSE;
```

```
    }
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);
    return TRUE;
}
```

首先定义窗口句柄 hWnd，以保存利用 CreateWindow()函数创建的窗口的返回句柄值。再将与窗口关联的实例句柄值赋给全局变量hInst。然后创建窗口，如果创建成功，则调用 ShowWindow()函数显示，否则返回 FALSE。

CreateWindow()函数的原型为：

```
HWND CreateWindow(
  LPCTSTR lpClassName,          // 指向注册过的窗口类名
  LPCTSTR lpWindowName,         // 指向窗口标题栏上显示的字符串
  DWORD dwStyle,                // 窗口风格
  int x,                        // 窗口左上角的水平坐标
  int y,                        // 窗口左上角的垂直坐标
  int nWidth,                   // 窗口宽度
  int nHeight,                  // 窗口高度
  HWND hWndParent,              // 指向父窗口的句柄，如果没有父窗口则为 NULL
  HMENU hMenu,                  // 窗口菜单句柄
  HANDLE hInstance,             // 程序实例句柄
  LPVOID lpParam                // 可忽略，通常为 NULL
);
```

至此主程序 WinMain()介绍完毕，那么程序与消息处理函数 WndProc 又是怎样联系起来的呢？请看 MyRegisterClass()函数：

```
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEX wcex;
    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.style = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc = (WNDPROC)WndProc;
    wcex.cbClsExtra = 0;
    wcex.cbWndExtra = 0;
    wcex.hInstance = hInstance;
    wcex.hIcon = LoadIcon(hInstance, (LPCTSTR)IDI_WIN32_INSTANCE);
    wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
    wcex.lpszMenuName = (LPCTSTR)IDC_WIN32_INSTANCE;
    wcex.lpszClassName = szWindowClass;
    wcex.hIconSm = LoadIcon(wcex.hInstance, (LPCTSTR)IDI_SMALL);
    return RegisterClassEx(&wcex);
}
```

注册窗口类时，将 WndProc（消息处理函数的名称）参数赋给了 wcex 的成员变量 lpfnWndProc，并且将全局字符串 szWindowClass 赋给了 wcex 的 lpszClassName，它将唯一地标识所要注册的类，以后要使用该类创建一个窗口，CreateWindow()函数的第二个参数应该设为 szWindowClass。

主函数 WinMain 是 Windows 程序框架的主要部分，由于 Windows 程序是基于消息处理的，所以消息处理函数也算作程序框架的一部分。说到消息，我觉得从基本的 C 编程转到 Windows 编程最难的一点可能就是理解 Windows 消息机制，虽然说是机制但不必觉得高深莫测，其实很简单，举个例子来说吧，在一个应用程序中有一个按钮，当按钮被按下时，我们已经习惯的程序会产生一个动作，在这样一个简单的过程中大部分事情是操作系统替我们完成了，检测按钮的动作，并将这个动作表现出来：按钮的按下、起来；按钮被按下时它会向它所在的程序发送一个消息“我被按下了”，程序中有一个消息处理函数捕捉到这个消息，接着转而处理它，也就是调用相关的函数。

接着看一下消息处理函数，消息处理函数其实是一个消息循环，是一个回调函数，什么是回调函数暂不用管它，只管理解为：函数由你定义而不归你调用，是由操作系统来调用的。下面来看一下，消息处理函数的全部代码：

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;

    switch (message)
    {
    case WM_COMMAND:
        wmId = LOWORD(wParam);
        wmEvent = HIWORD(wParam);
        // 分析菜单选择
        switch (wmId)
        {
        case IDM_ABOUT:
            DialogBox(hInst, (LPCTSTR)IDD_ABOUTBOX, hWnd, (DLGPROC)About);
            break;
        case IDM_EXIT:
            DestroyWindow(hWnd);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
        break;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        // TODO: 在此添加任意绘图代码
        EndPaint(hWnd, &ps);
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;
```

```
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

这个函数是系统默认的消息处理函数，也就是说，窗口程序的消息处理函数处理的只是我们感兴趣的消息，即上面的 case 语句，对于其他的消息还是使用系统默认的消息处理函数。消息处理函数是要在窗口主程序中登记的，登记为自定义的消息处理函数之后，系统的消息就由它来处理了。所以，在本函数的最后要使用该默认函数，否则其他的消息本程序就无法处理了。

比如一般在 WM_PAINT 中处理绘图，当关闭一个程序时，系统通常会发送一个 WM_DESTROY 消息给应用程序。

1.2 C++面向对象设计语言简介

C++就是使用面向对象（Object-Oriented，OO）思想提升了的 C 语言（尽管许多人都认为 C++和 C 语言是两门不同的语言，但我仍然趋向于认为 C++是 C 语言的扩展和升级），主要在以下几个方面进行了升级：

- 类的封装。
- 继承。
- 多态。

类是将数据和函数组合起来的一种方式。通常，使用 C 语言编程时，用数据结构来存储结构，用函数来处理这些数据，然而，使用 C++时，数据和处理数据的函数都封装在一个类中，这样做的好处是什么呢？这样可以将封装好的类看成有属性且具有某种行为的对象。这是一种更抽象更贴近实际情况的思考方式。

C++的另一项特性是继承。创建类后，便可以指定类对象之间的关系，在一个类的基础上派生出另一个类。现实世界本来就是这样的，例如有一种名为 Student 的类，它具有一些内部属性和特定的行为，如名字、性别等都可以成为它的属性；它可以具有一些行为（如去参加考试等）。但在这里，Student 是一个通用概念，比如小学生和大学生的行为显然是不一样的，而且小学生和大学生都具有他们特定的属性（比如小学生具有过儿童节的特权，大学生就没有这个权利了。）这时，就需要从 Student 类派生出两种新类，分别为 LStudent（小学生）和 HStudent（大学生）。

图 1-1 描述了 Student、LStudent 和 HStudent 之间的关系。明白这两个新类是如何从 Student 类派生出来的吗？LStudent 和 HStudent 拥有 Student 所拥有的所有属性、方法和行为，但它们又具有自身特有的属性和方法。这就是继承的基本概念：可以在已有类的基础上创建出更复杂的类。另外，还有多重继承，它让你能够以多个类为基础，创建出新的类。

C++和面向对象编程最重要的一点是多态（Polymorphism），其含义是“多种形式”。在 C++语境下，多态指的是根据不同的环境，函数和运算符有不同的功能。例如，在 C 语言中，表达式 a+b 表示将 a 和 b 相加，在这里 a 和 b 必须是内置类型变量，如 int、float、char、short 等。也就是在 C 语言中，不能定义一个新类型，然后将这种类型的变量 a 和 b 相加。然而在 C++中，完全可以重载+、-、*、/、[]等运算符，根据数据的类型执行不同的运算。

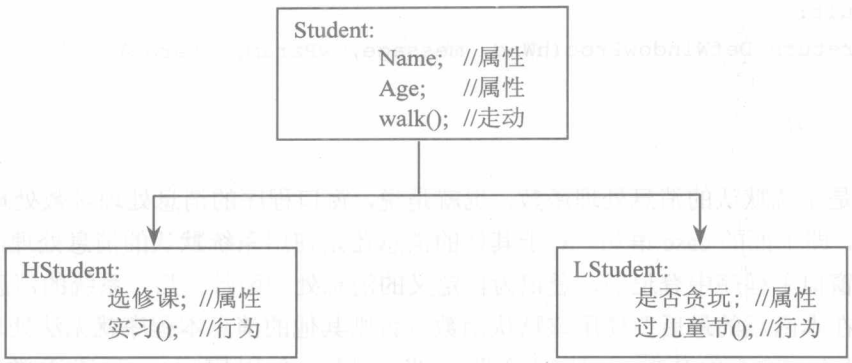

图 1-1 继承的关系

另外，也可以重载函数。例如，加上你编写的一个函数：

```
int Add(int a, int b)
{
    return a + b;
}
```

这个函数接受两个整数作为参数，如果提供的是浮点数，将转换成整数，然后传递给函数。将浮点数转换成整数的过程中，降低了数据的精度。为了避免这种情况，在 C++中，可以重载上述函数：

```
float Add(float a, float b)
{
    return a + b;
}
```

虽然这些函数的名称相同，但它们接受不同类型的参数。在编译器看来，它们是完全不同的函数，使用整数作为参数时，将调用第一个函数；使用浮点数作为参数时，将调用第二个函数。如果使用一个整数和一个浮点数作为参数，问题将变得复杂。编程时应尽量避免这种情况的发生。

以上是 C++的主要特性。当然，相对 C 语言来说，新增了大量与此相关的语法和规则，但从很大程度上来说，C++的一切都与这 3 个新概念有关。

下面详细介绍 C++中引入的新特性和新概念。

1.2.1 流式 I/O

C 语言中普遍采用 printf 和 scanf 函数来处理键盘的输入输出，类似地采用 fprintf 和 fscanf 函数来处理文件的输入输出。然而 printf()的缺点是，格式指定符太多，如%d、%x、%f 等，不便于记忆。另外，scanf()更糟糕，如果忘记了传入的参数应该是参数的地址，情况将变得一团糟。例如：

```
int x;
scanf("%d", x);
```

上述代码是错误的，应传入 x 的地址，即&x。很多人都犯过这样的错误，唯一不需要使用地址符的情况是使用字符串时，因为字符串变量本来就是地址。这就是 C++创建新的 I/O stream 类的原因。它能够识别变量的类型，不需要显式指定。这个类是在头文件<iostream.h>中定义的，要在 C++中使用这个类的函数，必须包含这个函数。这样，就可以访问流 cin、cout、cerr 等了。