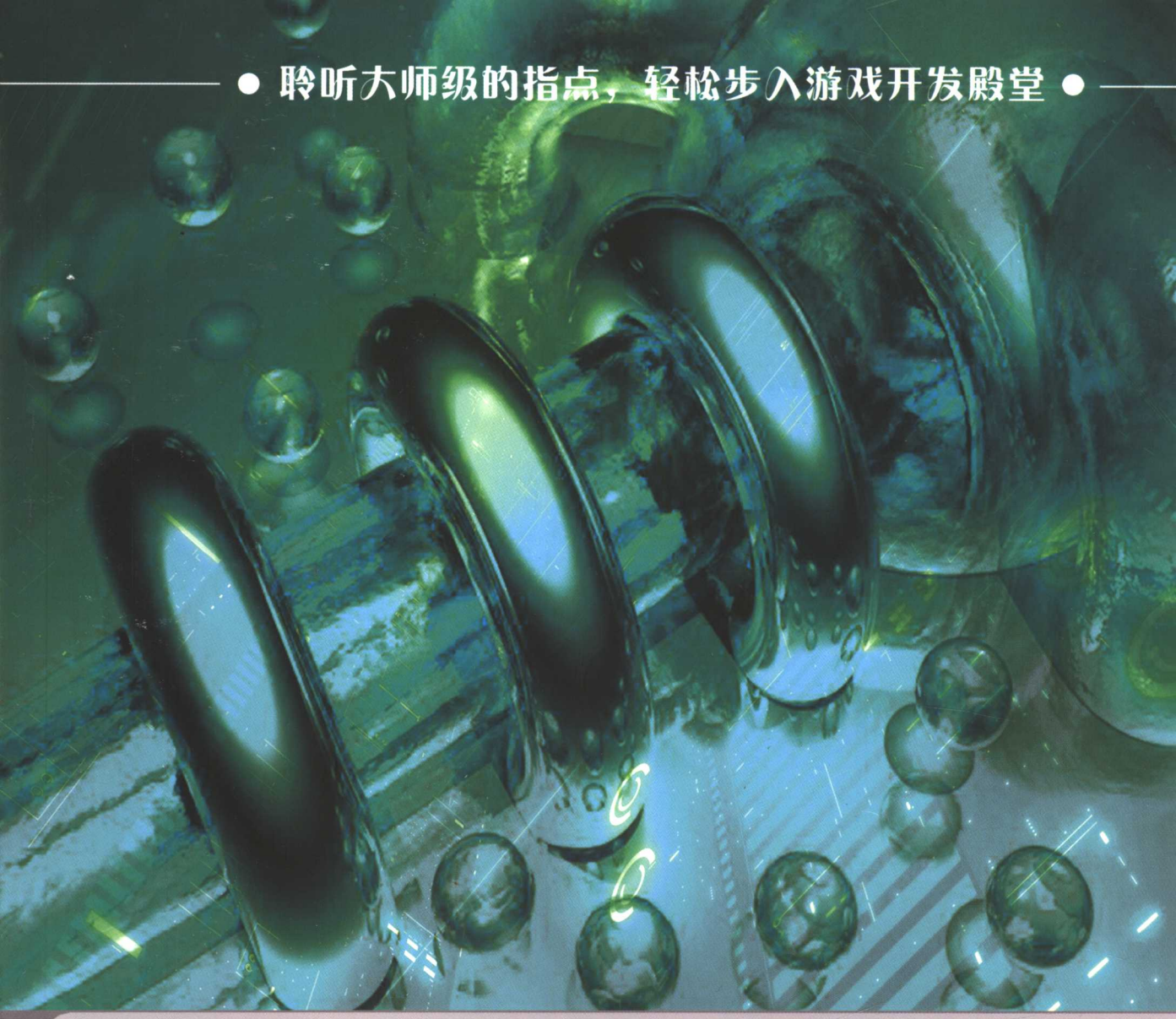
聆听大师级的指点，轻松步入游戏开发殿堂



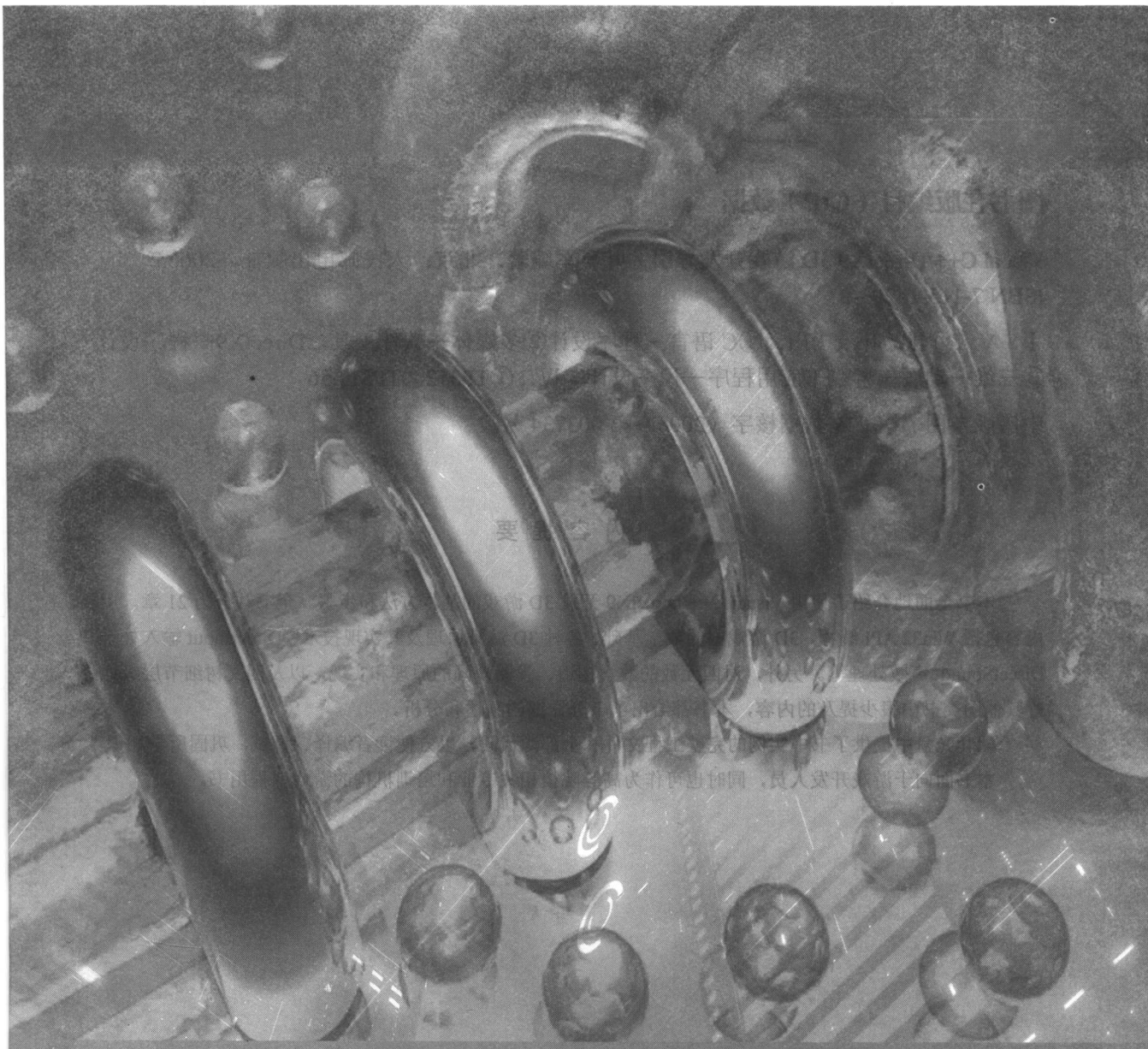
Visual C++/DirectX9 3D游戏开发导引

GAME PROGRAMMING TUTORIALS

叶至军 编著

附源代码
CD-ROM

人民邮电出版社
POSTS & TELECOM PRESS



Visual C++/DirectX9 3D游戏开发导引

GAME PROGRAMMING TUTORIALS

叶至军 编著

人民邮电出版社
POSTS & TELECOM PRESS

图书在版编目（CIP）数据

Visual C++/DirectX9 3D 游戏开发导引 / 叶至军编著. 北京：人民邮电出版社，2006.2
ISBN 7-115-14293-9

Ⅰ. V... Ⅱ. 叶... Ⅲ. ①C 语言—程序设计②多媒体—软件工具，DirectX9—程序设计③三维—动画—游戏—应用程序—程序设计 Ⅳ. ①TP312②TP311.56

中国版本图书馆 CIP 数据核字（2006）第 003734 号

内 容 提 要

本书全面地介绍了使用 Visual C++/DirectX9 开发 3D 游戏的基本方法和步骤。全书共分为 21 章，主要内容包括 Win32 API 编程、3D 游戏中的数学基础、各种 3D 渲染原理及其实现技术、DirectInput 输入处理和 DirectSound 声效处理等。另外，对四元数的数学构造、骨骼动画的原理和实现，以及四叉树细节层级地形等其他同类书中很少提及的内容，本书都给出了十分详细的阐述和分析。

配套光盘中提供了书中实例的完整工程文件和全部源代码，以方便读者编译、调试，巩固所学知识。

本书面向于游戏开发人员，同时也可作为高等院校相关专业和培训机构的游戏课程用书。

Visual C++/DirectX9 3D 游戏开发导引

◆ 编　　著　叶至军
责任编辑　汤　倩

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：30.5
字数：744 千字　　2006 年 2 月第 1 版
印数：1 – 5 000 册　　2006 年 2 月北京第 1 次印刷

ISBN 7-115-14293-9/TP • 5151

定价：54.00 元（附光盘）

读者服务热线：(010)67132692　印装质量热线：(010)67129223

前 言

近年来，随着游戏玩家人数的增多，游戏市场规模不断扩大，游戏产业成为了备受关注的产业之一。对游戏的策划、美术、程序开发、市场拓展和技术维护等方面的人才需求也逐渐增多。虽然国内的程序开发人员较多，但主要以开发企业商用软件为主，而游戏开发人员却极度匮乏，无法满足游戏产业对开发人员的需求。

3D 游戏是当前游戏的主流，其核心技术是 3D 图形的渲染。一般来说，从事 3D 游戏开发不外乎两种方法，一是利用已有的具有世界级水平的 3D 引擎进行开发；另一种是自行开发 3D 引擎。前者因为容易上手，成为大多数游戏公司的选择；后者则对图形渲染算法有更高的要求。实际上真正属于原创的引擎很少，但也不乏优秀的开源引擎，如 OGRE 引擎和 NeL 引擎等。

使用或开发 3D 引擎，需要熟悉 3D 图形显示原理、DirectX 或 OpenGL 图形 API 的使用，以及相关的数学、物理方法。因此，本书全面地介绍了使用 DirectX 9.0 提供的图形编程接口进行 3D 游戏开发的基本方法。考虑到游戏运行速度的问题，书中采用 Win32 API 编程模式而不是 MFC 类库来开发所有的示例程序。

本书的特色主要体现在以下几个方面。

（1）对书中内容的规划具有较强的逻辑性，从 C/C++基础回顾→Win32 API 编程→3D 游戏的数学基础→3D 渲染流程的原理等引导性的内容，到 3D 的各种渲染技术，全面、清晰地介绍了各种技术的实现方法，同时，也详细剖析了技术背后涉及的各种数学原理。

（2）为了使读者对游戏引擎的开发有一个初步的认识，书中开发了若干示例性的引擎类，并利用这些引擎类来开发游戏程序。

（3）对于 3D 游戏开发中涉及的热点问题和技术，如四元数的纯数学构造分析、利用 DirectX 9.0 提供的接口实现骨骼动画，以及视点相关的细节层级四叉树地形等其他同类书中很少提及的内容，本书都有详尽地介绍。

全书共分 21 章，下面是各章的内容简要介绍。

第 1 章“C/C++基础回顾”，论述编程行为，并解析了 C/C++的程序结构、C++类的模块以及 C++的常用语言要素。

第 2 章“Windows API 编程基础”，介绍 Windows 程序执行原理、消息机制、窗口回调

过程函数和 Win32 编程。

第 3 章“DirectX 3D 游戏开发入门”，介绍 3D 渲染管道流水线的渲染流程，世界坐标系、摄影坐标系、透视投影、COM 组件技术和 DirectX 9.0 的安装和配置。

第 4 章“3D 游戏中的数学基础”，介绍向量、矩阵、四元数、坐标变换等重要内容。

第 5 章“键盘、鼠标和游戏杆的输入处理”，介绍使用 DirectInput 接口读取键盘、鼠标和游戏杆输入的方法。

第 6 章“游戏框架、输入和时钟引擎类”，将 Windows 程序的架构和输入处理代码用类包装，并提供一个获取精确时钟和帧频 FPS 的时钟类。

第 7 章“基本三角形面的绘制”，介绍 DirectX 图形渲染架构、D3D 接口和 D3D 设备和顶点缓冲区的创建。以简单的三角形面为例，介绍渲染管道流水线上的顶点的渲染。

第 8 章“基本立体面的绘制”，深入介绍 3D 的三角形面渲染方式、顶点索引缓冲区、场景物体的放置、取景、透视投影以及视口的设置。

第 9 章“材质和光照处理”，介绍场景的光照处理和场景物体对于光照反射方面的材质属性设置，以及顶点的法向量计算。

第 10 章“纹理贴图”，介绍纹理接口对象的创建和纹理映射物体表面的相关内容。

第 11 章“Alpha 颜色混合处理”，介绍物体表面的多种颜色融合处理技术。

第 12 章“XFile 网格的应用”，介绍.x 文件的基本格式，以及利用.x 文件提供的顶点网格数据进行 3D 物体表面渲染的方法。

第 13 章“骨骼动画的实现”，深入介绍.x 文件的动画格式、骨骼动画原理和骨骼蒙皮的渲染处理。

第 14 章“视点相关的 LOD 地形渲染”，用四叉树的地形分割方法，动态生成具有不同细节层级的地形三角形顶点数据，为地形渲染提速。

第 15 章“基本 3D 渲染引擎类”，将前面各章的 3D 渲染代码进行类的封装，给出渲染类，用来简化后面章节的例程代码。

第 16 章“天空和广告牌技术”，介绍矩形面天空和更为真实的圆形天空的渲染方法，以及 2Din3D 的二维图形仿真处理。

第 17 章“雾化处理”，在场景中添加雾化特效，介绍顶点雾化和像素雾化两种处理方式。

第 18 章“文字和能量格渲染”，介绍场景的文字显示和角色能量格的渲染。

第 19 章“粒子系统”，介绍使用粒子实现场景爆炸效果的渲染。

第 20 章“Stencil 蒙板阴影处理”，介绍平面阴影的顶点计算和渲染方法，以及 Stencil 蒙板缓冲区在阴影处理中的应用，以及当前流行的 ShadowMap 和 ShadowVolume 阴影算法。

第 21 章“DirectSound 声音播放”，介绍 DirectSound 的声音播放技术，包括声音的播放原理、主次缓冲区的获取、声音数据的装入、控制和播放等。

本书主要由叶至军编写，其他参与编写的人员有欧阳国、孙德俊、郭景春、苏骞、叶至力、李德扬和叶青峰等。焦焦、符译尹和符传学对本书的顺利完稿给予了大力支持，在此表示感谢。在编写过程中，我们力求精益求精，但由于水平有限，书中难免存在不足之处，恳请读者批评指正。如有任何问题或意见，可以发 E-mail 到 tangqian@ptpress.com.cn 与我们联系。

编　者

光盘使用说明

为了方便读者调试程序和观察程序的执行结果，本书附带了一张光盘，提供全书所有程序的工程文件。下面简要介绍光盘的使用方法。

一、光盘的运行环境

硬件环境：CPU 的主频在 500MHz 以上、内存 128MB 以上，要求配置 3D 显示卡。

软件平台：操作系统为 Windows 2000/XP，集成开发平台为 Visual C++ 6.0，需要下载并正确安装 DirectX 9.0，并确保 Windows 已有的旧版 DirectX 完全清除掉（否则可能会导致有些程序无法运行）。

二、光盘目录

图 1 所示是光盘根目录的内容，包括各章的程序目录（第 15 章无例程）和 GameEngine 目录。

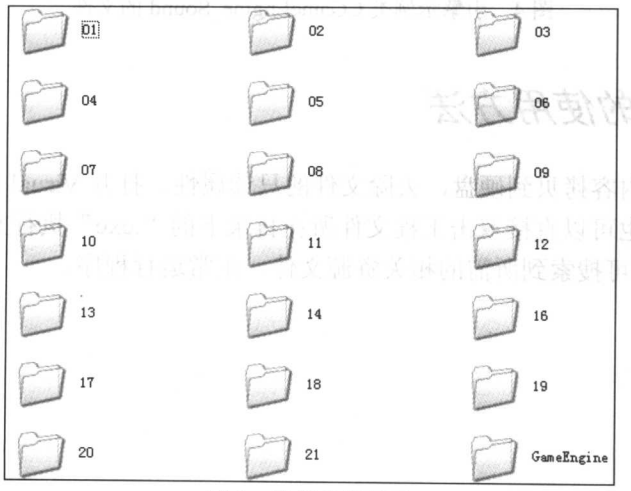


图 1　光盘的根目录

图 2 所示举例说明了程序目录的内容，其中包括工程文件、程序执行文件和程序截图等。

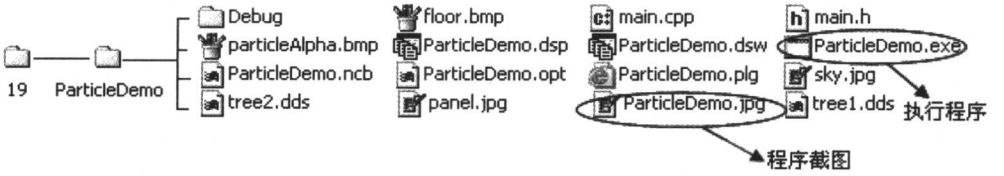


图 2　第 19 章的子目录

图 3 所示是“GameEngine”目录的内容，包括本书开发的所有示例性引擎类。

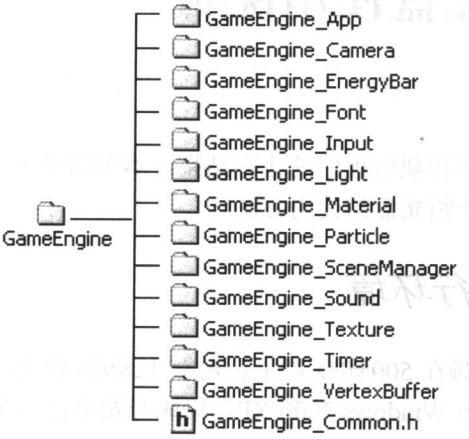


图 3　GameEngine 的子目录

图 4 所示是引擎类目录中包含的内容，包括类的.h 文件和.cpp 文件。

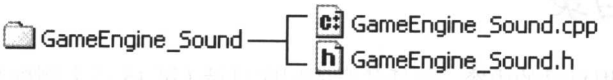


图 4　引擎示例类 CGameEngine_Sound 的文件

三、光盘的使用方法

建议将光盘的内容拷贝到硬盘，去除文件的只读属性。打开 Visual C++工程文件，进行程序的运行调试。也可以直接双击工程文件所在目录下的“.exe”执行文件（实际从 Debug 目录拷贝出来），即可搜索到所需的相关资源文件，正常运行程序。

目 录

第 1 章 C/C++基础回顾

C++从 C 扩充而来并保持兼容，C++的编译器（例如微软的 Visual C++）仍可继续编译 C 的程序。C++继承了 C 作为中级语言所具有的对硬件的位、字节和地址等进行操作的能力，并提供了类、继承、重载、覆盖、接口和模板等面向对象语言处理，有效地实现了程序数据和代码的组织，相当程度上解决了 C 模块开发中数据容易紊乱的问题。在中大型软件项目开发中，C++从开发的时间和费用，以及软件代码的可靠、可重用、可扩充和可维护等方面，都显示了极其强大的优越性。目前，C++已成为 3D 游戏开发的首选语言。

C++与 C 密切相关，很多 C++的面向对象语言元素，都是针对 C 程序开发所暴露的问题而提出的，并不是纯粹的语言理论创造。因此，不能脱离 C 去学习 C++，况且 C 还是 C++的一个子集。正如不可能不清楚硬件和操作系统细节而去期盼编写高质量的系统程序。

C++的语言要素十分庞杂，当然也没有必要全面掌握。正如使用 UNIX/Linux/DOS 系统，没有必要全部掌握它的命令，基础的内容完全可以在需要的时候再去研究学习。因此，本章对 C/C++的一些最基本的用法给出简要的综述，以备后续章节所用。

1.1 浅谈编程行为

如果现在仅有机器码，程序员直接使用 0 和 1 的指令串进行编程，那么，从某种意义上来说，只要清楚了解相关硬件寄存器的作用和指令意义，就可实行算法编程，不需要太长时间的语言学习。但换来的是，编程工作量大，效率低，具有高度的指令抽象性，不易记忆，程序查误和排错，需要花费更多的时间。

现在各种高级的编程语言，如 C++、Java 和 PowerBuilder 等，一般都是用于编写程序代码（源码），然后调用编译器进行编译和汇编，产生可由计算机硬件执行的机器码。从以上的编程过程来看，应用高级语言进行编程，相当于程序员与隐蔽在某处的编译器精灵的对话，

一次性地将程序源代码文本，提交编译器精灵，让它进行翻译，转变为 0 和 1 组成的机器码，作为指令提交给机器执行。

由此可见，程序语言（包括使用助记符的汇编语言）的一个最根本的特性，就是能够充分利用符号帮助思考，一方面将机器指令符号化，另一方面将日积月累下来的各种功能指令块，通过某种符号形式（即函数）支撑这些指令码集。最终把符号转化为机器码，致使编译和汇编链接技术被引入和应用。

如图 1-1 所示，是一种假想的十分简单的程序机器码的观察模型，连续内存中，每个字节具有一个地址编号，数据区中的数据可占用 1 个字节或 2 个字节不等，代码区中的指令则按照数据的实际存放方式，进行读取（或写入）加工，生成新的数据进行输出。

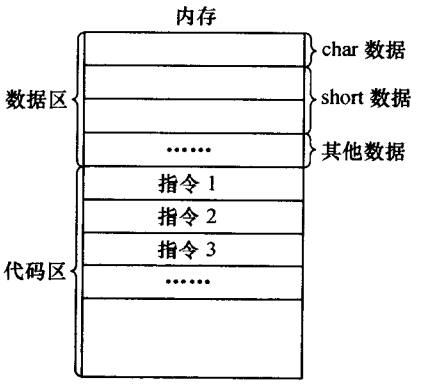


图 1-1　一种程序机器码的观察模型

对于高级语言的编程，变量的声明，相当于是让编译器在图 1-1 所示的数据区中分配一个数据空间，变量类型确定了所占字节的多少，例如，char 和 short 的 C 语言数据类型，将分别占 1 个和 2 个字节。与汇编语言不同，变量名将指称整个变量所占据的内存字节空间，并不是所占据空间的首地址。在程序代码中，若需要确定变量所在地址，可通过一个“&”取地址操作来完成。

变量的初始化赋值，相当于将具体的数据填写入该变量所占的字节空间中，否则变量空间仍保留某程序在该处执行后的“残余”值，这个值具有不确定性。不同的变量类型，具有相应的 0 和 1 编码格式。

直观地说，高级语言中程序的变量声明和初始化赋值，最终是让编译器完成图 1-1 所示的数据区数据布局。早期的 C 语言版本，就有一个现在已废弃了的规定，变量的声明语句必须放在程序的首行代码前面。

相当于机器指令的程序代码，普遍采用函数的形式来书写。任何一种高级语言，自身都会提供一些最底层的函数供程序员使用。尤其是与操作系统密切相关的 C 语言，将系统调用的指令用函数的形式提供出来，使语言具有最基本的与硬件和操作系统的交互操作功能，如文件 IO 读写功能等。其他从 C 衍生出来的语言，有针对性地进行简化和包装，提供新的语言函数调用形式。

在语言函数的基础上，程序员将开发出更多的应用于具体软件项目的函数。函数内蕴了执行指令，这些指令构成了一个指令集合。将不同函数对应的指令集合并起来，就可组成更

大的指令集，从而最终由编译器翻译生成图 1-1 所示的程序指令系列。由此可见，函数起到最基本的模块分解作用，也就是说，一个程序的开发，可将归结为一系列函数的编写，最后将函数调用起来，就可完成。

尽管程序语言的“函数”一词借用了数学上的函数说法，但是更应该从语言的角度，去理解和使用函数。例如，C 语言中计算自然对数的 double log(double num)函数，它的一个具体调用 log(3.0f)语句，就是告诉编译器，将计算 log(3.0)的指令码集，“嵌入”正在构造的程序代码中。实际的编译处理是使用一条指针，转向 log 的指令码处。对于这一点细节，程序员在开发过程中是不必深究的。

在代码中调用一条条的函数，相当于在图 1-1 所示的代码区中一批批地写入将要执行的指令。

了解了语言必备的变量声明和函数调用功能之后，接下来看看语言的结构成分——语句结构化和数据结构化。

语句结构化是指语言所提供的一些结构化的流程控制语句，例如 C 语言的 while、for 和 if 等，这些流控制实际就是利用比较指令和跳转指令，将当前程序计数器的值，调整到另一地址处执行（或循环执行）。使用这些流程控制语句，可使程序内部呈现模块的效果，不必再使用类似于 goto 的原始跳转语句，将执行地址转移到“千里之外”，而难以在事后进行程序代码的阅读和分析。

数据结构化，体现了语言为适应实际开发而提供的一种无限扩展数据类型的能力，它允许将一些原子数据类型（如 C 的 char、short、int、long 和 float 等），通过某种语言的数据结构成分（如 C 语言的结构体定义），组合生成新数据类型。对于较大规模的数据处理来说，数据的存储方法在一定程度上影响了算法的制定，两者相辅相成，相得益彰。结构化的数据类型，就是让编译器在程序数据区中完成所期待的数据布局。

在程序编写完毕后，需要进行程序代码的错误检查。对于直接使用机器指令编写的程序，错误检查主要包括算法逻辑上的错误和人为的指令数据书写错误。发展到高级语言，后一种类型的错误，可自动由编译器在程序源码编译成汇编码之前检查出来。

编译器的这种错误检查，在判断每一语句是否存在错误时，是依据编译器所编译的语言的语法规则给出的。每一条语法规则的制定，一方面是技术上和阅读上的要求，例如，语句的分割符号，是为了使编译器知道一条语句的结束和另一条语句的开始。函数体的大括号，是为了表明内部为函数代码，也可指示函数代码的开始和结束。另一方面，是针对一些易出现的错误而提出的，使错误在编译之前就发现出来，而不致于隐匿在机器代码中，直到某一次的执行，才暴露出来而引发事故。例如，为了避免变量的重复命名，给变量的访问添加了一些全局、局部和 static 性质的变量声明，使变量具有一个可见区域，以限制某些区域的代码访问变量。又例如，变量赋值的类型合法性检查，可有效地防止数据的人为书写错误。

对编程的行为做一个概括，就是程序员与编译器精灵的对话过程，只是对话的模式采用文件，而不是逐句式的即时交流，对话的语言为高级编程语言，而编译器则是与计算机硬件的对话，它也是一次性地提交机器码的对话文本，让机器执行文本指令。

程序员编写的程序语句，必须符合该语言的语法规则，否则无法让编译器理解并完成编译。在编写程序的过程中，可借助图 1-1 的理想模型，考量每一条代码语句的作用，或者是数据空间作用，或者是代码指令作用。

以上完全是从实用角度对编程行为所做的一些反思，无意引申到更为一般的哲学层面。