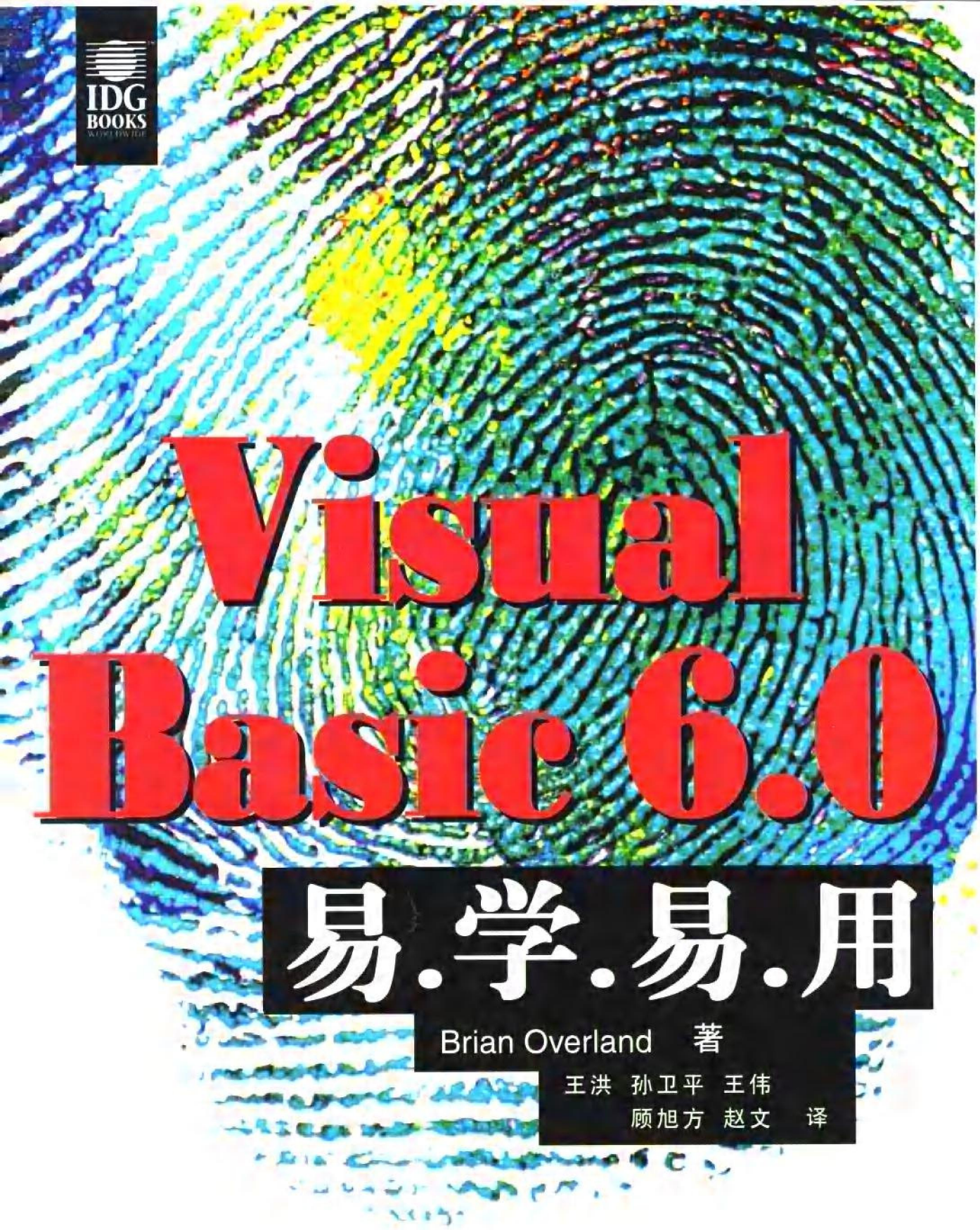

Visual Basic 6.0

易.学.易.用

Brian Overland 著

王洪 孙卫平 王伟
顾旭方 赵文 译

人民邮电出版社

Visual Basic 6.0 易学易用

Brian Overland 著

王 洪 孙卫平 王 伟 顾旭方 赵 文 译

人民邮电出版社

内 容 提 要

Visual Basic 6.0 是一种功能强大、使用灵活的应用程序开发工具，本书从程序开发和语言指南的角度对它进行了详细介绍。全书共 22 章，内容涵盖了 Visual Basic 6.0 发展史、基础知识、编程技巧、各种数据类型应用、高级控件操作、面向对象编程、数据库应用、ActiveX 控件、图形绘制、文件操作、代码调试及各种语句函数的分类描述等等。

本书内容丰富，叙述详细，实用性强，适合广大程序员及计算机爱好者阅读。

Visual Basic 6.0 易学易用

◆ 著　Brian Overland
译　王 洪　孙卫平　王 伟　顾旭方　赵 文
责任编辑　肖学云

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
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：21.5
字数：538 千字　　1999 年 5 月第 1 版
印数：1 – 6 000 册　　1999 年 5 月北京第 1 次印刷
著作权合同登记 图字：01 – 98 – 2314 号
ISBN 7-115-07592-1/TP·972

定价：32.00 元

版 权 声 明

前言

欢迎阅读《Visual Basic 6.0 易学易用》，它不仅是学习 Visual Basic 6.0(以下简称 VB6.0)的指南，更是一本通常意义上的手册。本书也可以看成是两本书，它既是一本可供初、中、高级程序员等使用的简洁、详细的参考书，又是一本难度适中的经常被人们讨论的课题的概述。本书提供了查询资料的许多方法，同时，对于常见的问题本书也提供了简单明了的答案，如果你想要一本放在你键盘旁边，或者能在公共汽车上阅读的编程书籍，本书正是这样的一本书。

正如书名中所隐含的意义，本书还有一个特别的任务：澄清一些使人非常困惑的编程概念。该书还非常侧重于一些富于挑战性的问题，如：面向对象的编程、ActiveX 控件、低级的图形绘制和高级的文件操作。本书的目标是不仅解释如何使用这些新的特点，而且还讲解如何利用它们来为你服务，也就是它们为什么要这样做。

正如书名所指出的，对 VB6.0 的新特性要给予足够的注意。许多章节都是以"VB6.0 中的新特性"这部分开始，以便使你对每一个主题中的新东西有一个快速的提纲挈领的了解。

我从事 VB 方面的工作已经很长时间了，我是最初的 VB1.0 的项目负责人之一，虽然我使用过许多别的语言，但对 VB 还是怀有很深的感情。你可以获得关于这个项目在设计和发展过程中的诸如"为什么"和"怎么样"之类问题的一些回答。总之，设计 Visual Basic 的目的是有乐趣地使用它，我相信你会拥有我每次坐下来去编程所获得的快乐。

谁可以使用这本书

正如前面所提到的，这本书不仅是一本对所有 VB 程序员都有用的参考资料，它还讨论了一些难度适中的课题。不过，本书首先会回顾基本知识，而且高级程序员也能从对 VB 的新特点的讨论中受益匪浅。

虽然本书也回顾了基础知识，但你最好还是在读完一本完全面向初学者傻瓜书以后，再来阅读这本书。如果你有以前版本的 Basic 语言或者像 C++之类的语言的背景知识，这本书将会使你快速掌握 VB6.0。

为了编辑和运行例子程序，你需要一份 VB6.0 的拷贝，VB5.0 支持大部分例程，但不是全部。一些例程使用了 6.0 版本中的新特性。

如何使用这本书

本书分为两部分，这两部分相辅相成，互为参考。

第一部分：掌握 VB 6.0（第 1～12 章）

在这部分中能了解所讨论的问题的概况，如控件、面向对象或数据库操作等。在这部分中也能获得一个总结，该总结概括了 VB6.0 带给每一领域的新特点。每一章自成一节课程，首先讲述一般性的技术，然后再引导你从头到尾学完简单而有用的例程。文字部分用于分析每一个例子，解释代码的每一部分如何做以及为什么这样做，这部分也具有丰富的插图和概念示意图，这也与 VB 就应该是可视的风格的观点是一致的。

第一部分以一个专门的总结性的章节作为结束，该章概括了在前面章节所采用的技术，也概括了一些其它技术。本书是一本采用简短、直接、渐进的教学方式来指导你使用 Visual Basic 完成任务的指南。

第二部分：表和参考数据（第 13 章～结束）

尽管在这样狭小的篇幅里总结 VB 的全部方面是不可能的，但第二部分概括了语言本身的核心部分，包括面向对象的新特点和调试命令。这部分也对 VB6.0 中新的内嵌函数给予了重视，给出了在工具箱中的每一个标准控件的概览，说明了它们中的关键属性、方法和事件。控件概览能够节省你在查询每个控件所支持的很长的属性列表时所用的时间。

使用约定

为快捷起见，在全书中使用了一些约定，采用这些约定的目的是以较少的时间和篇幅去说明更多的内容。

源代码

例子中的代码使用特殊的字体与正文进行区分：

```
pi = Atn(1)*4
angle1 = 135*pi/180
angle2 = 45*pi/180
Circle(2000,2000),1000, ,angle1,angle2
```

正文中的关键字和示例

出现在正文中的关键字、命令名和代码片段都采用一种特殊的关键字字体。仅仅举一个例子，使用这种字体能够将关键字 If 与正文中的单词 if 区分开来。

表达语法

本书中经常给出了你所使用的一条特定的语句或函数所包含的所有信息（或几乎所有的信息），为简明清楚地表达这些信息，本书采用了与在 VB 环境中所采用的相同的语法形式。关键字和标点使用黑体，占位符（要填入值的表达项）使用斜体。

例如，下列语法显示了你必须像所示的那样准确键入 Dim 一词以及你自己采用的变量名：

Dim *variable_name*

方括号表示了一个可选的项目，你可以选用它，也可以不用。上述语句的一个更完整的语法表达方式是：

Dim *variable_name* [**As** *type*]

这表示了可选项目有 As 关键字和 type 占位符，因为它们都在同一个方括号里面，你必须同时使用它们。

最后，你还将看到省略号(...)的用法，这表示了一个表达项可以被重复任意次。例如 在 If ...Then 语法表示中你可以包含任意个从句，每一个 ElseIf 从句都必须有它自己的条件表达式和 Then 关键字：

If *condition* **Then**
    statements
[**ElseIf** *condition* **Then**
    statements]...
[**Else**
    statements]
End If

注释符

为简单起见，还有一个专门的符号：注释符。注释符中的内容不在所要讨论的部分之中，但需要你以后注意。注释符经常给出你可能会忘记的重要警示或技巧，有时注释符也会指出 VB6.0 中的一个新特性。

注释：这就是一个注释符的例子。

与作者联系

欢迎你回信并提出意见，对某些人提出的技术问题我不总是能很快地回答，但我会尝试着去做，我特别欢迎你们回信谈谈使用这本书的体会，以及提出建设性的意见以使今后出版的书更完善。我的 email 地址是 Briano2u@aol.com。

目 录

第一章 VB 的历史 …… 1
1.1 初始阶段 …… 1
1.2 理想成真 …… 2
1.3 创造两者之间的结合 …… 3
1.4 关于名字的由来 …… 5
1.5 定制控件免除你的担忧 …… 5
1.6 不断前进，不断提高 …… 6
1.7 Visual Basic 的下一个版本 …… 6
1.8 继续前进：从 3.0 版到 5.0 版 …… 7
1.9 新版 6.0 …… 8
1.10 Basic 走过了一段漫长的历程 …… 9
第二章 开始使用 VB …… 11
2.1 VB6.0 中的新特性 …… 11
2.2 启动 Visual Basic …… 11
2.3 开始起步：绘制控件 …… 13
2.4 设置初始属性值 …… 15
2.5 编写代码：进入事件 …… 17
2.6 编写代码：设置属性值 …… 18
2.7 存储和管理工程 …… 21
2.8 另一个应用程序：专家解惑 …… 24
2.9 优化提高：关于事件的更多信息 …… 29
2.10 Visual Basic 开发总结 …… 30
第三章 编程提示与技巧 …… 32
3.1 VB6.0 中的新特性 …… 32
3.2 代码窗口管理 …… 32
3.3 代码输入中的快捷方式 …… 35
3.4 注释命令和注释块命令 …… 37
3.5 快速输入输出 …… 38
3.6 声明和使用变量 …… 39
3.7 提高自己：定义新的过程 …… 44
3.8 使用控件和对象的快捷方法 …… 47

第四章　图形化的 VB …… 50
4.1　VB6.0 中的新特性 …… 50
4.2　图形的一些基本知识 …… 50
4.3　使用图形方法绘制图形 …… 56
4.4　轻型图形控件 …… 61
4.5　示例程序：吹泡泡 …… 62
4.6　具有动画能力的 Image 控件 …… 64
4.7　加载和存储图像：Scribble 程序 …… 67
第五章　数组 …… 75
5.1　VB6.0 的新特性 …… 75
5.2　基本课程：数组 101 …… 75
5.3　动态数组 …… 79
5.4　二维数组和多维数组 …… 81
5.5　传递和返回数组 …… 83
5.6　参数数组 …… 85
5.7　控件数组 …… 86
5.8　本章中的关键字总结 …… 91
第六章　高级控件 …… 93
6.1　本章用到的控件 …… 93
6.2　滚动条控件 …… 94
6.3　文本框控件 …… 96
6.4　列表框控件及组合框控件 …… 100
6.5　RichTextBox 控件 …… 105
第七章　文件操作 …… 112
7.1　VB6.0 的新特点 …… 112
7.2　公共对话框控件 …… 112
7.3　一个简单应用：文本文件编辑器 …… 116
7.4　文件系统控件 …… 119
7.5　简单的应用程序：文件浏览器 …… 123
第八章　Basic 的面向对象编程 …… 126
8.1　VB 6.0 的新特性 …… 126
8.2　对象 …… 126
8.3　对象参照类 …… 128
8.4　创建对象 …… 129
8.5　在工程中定义一个新类 …… 134
8.6　高级属性声明 …… 139
8.7　对 For Each 语句的支持 …… 140
8.8　VB 面向对象的高级特性 …… 141
8.9　其它类型的类 …… 146

第九章　编写控件 …… 147
9.1　VB 6.0 的新特性 …… 147
9.2　开始使用 ActiveX 控件 …… 147
9.3　第一个 ActiveX 控件：BigX …… 148
9.4　轻型控件和透明性 …… 150
9.5　添加 Click 事件 …… 151
9.6　添加自定义事件 …… 152
9.7　另一个例子：微笑 …… 154
第十章　控件与属性 …… 160
10.1　添加一个简单的属性 …… 160
10.2　控件与属性的原理 …… 163
10.3　颜色、图形和其它特殊属性 …… 164
10.4　其它有用的 UserControl 事件 …… 172
10.5　绘制图标 …… 173
10.6　使用向导 …… 174
10.7　关于自定义控件的一些感想 …… 175
第十一章　数据库接口 …… 177
11.1　VB6.0 的新特性 …… 177
11.2　向工程中添加 ADO 数据控件 …… 178
11.3　创建一个简单的数据库接口 …… 179
11.4　通过设置控件属性连接数据库 …… 180
11.5　ADO 数据控件的实际应用 …… 184
11.6　用 Find 方法查找记录集 …… 185
11.7　使用向导 …… 187
11.8　其它的 ADO 类 …… 189
第十二章　问题集锦 …… 190
12.1　基本部分 …… 191
12.2　图形部分 …… 194
12.3　数组部分 …… 197
12.4　文件操作部分 …… 200
12.5　多窗体部分 …… 204
12.6　面向对象编程 …… 207
12.7　用户自定义控件（ActiveX） …… 210
第十三章　数据类型 …… 215
13.1　数据类型综述 …… 215
13.2　数据类型描述 …… 215
13.3　自动数据类型变换 …… 219
13.4　显式数据类型变换 …… 220
13.5　隐式变量声明 …… 220

第十四章　运算符 …… 222
14.1　概述 …… 222
14.2　高级运算符 …… 223
第十五章　标准控件 …… 227
第十六章　用对象编程 …… 241
16.1　对象变量 …… 241
16.2　类模块 …… 244
16.3　测试对象类型 …… 248
16.4　对象浏览器 …… 249
16.5　接口 …… 249
16.6　Collection 类对象 …… 253
16.7　Dictionary 类对象 …… 254
第十七章　控制结构 …… 256
第十八章　文件系统 …… 270
18.1　通用文件 I/O …… 270
18.2　顺序文本模式（Input，Output，Append） …… 273
18.3　随机和二进制模式 …… 275
18.4　通用文件命令 …… 278
18.5　FileSystemObject 模式 …… 279
第十九章　通用输入/输出 …… 286
第二十章　字符串处理功能 …… 294
第二十一章　数学运算功能 …… 309
第二十二章　调试命令 …… 315
22.1　通用的调试特征 …… 315
22.2　Debug 菜单上的命令 …… 318
22.3　View 菜单上的调试命令 …… 322
22.4　调试与事件流 …… 325
22.5　最后的感想 …… 325
附录 A　计算器例程代码 …… 327
附录 B　堆栈类 …… 330
附录 C　ASCII 字符表 …… 332

第一章 VB的历史

迄今为止所开发的所有语言，还没有哪一种能够使得使用它们的每一个人都成为程序员，但是 VB 却最有可能接近并实现这一点。当我注意到最近几年加入到 VB 中的新特性时，有时不禁喃喃自语"它不可能是如此容易吧！"。

当然，VB 对我来说似乎很容易，因为我自己写的第一个程序就是第一个 VB 版本中的程序，直到现在还在新的语言上运行着的程序。我很熟悉 Visual Basic ，我从来不必担心 object.property 这种奇怪的语法是什么意思，因为我是从商业界领先的开发者那里学到的，但是对于许多人—— 一些非常聪明的人来说，还是会在开始时遇到一点障碍。

后来，有了越来越多的对象的应用,其部分目的是使 Visual Basic 成为一种艺术形式的计算机语言，但其更重要的目的是支持微软即将推出的系统结构，这种结构的目标在于营造一个统一的环境，使得在其中的应用程序可以彼此交流，并且 VB 能够运行所有的应用程序。让我们回到这个项目的最初版本吧。

1.1 初始阶段

一开始有一个名叫 Bill Gates 的人和一个名为 Basic(或按当时叫 BASIC)的语言。到现在，盖茨发明的 Basic 并不比亨利·福特发明过的汽车多多少，他所做的工作是使 VB 能够为成千上万的人所使用，尽管微软有许多其它里程碑性的产品，但 VB 从某种意义上说是这家公司的基础，它带来的收益帮助了年轻的合作者们从创建它所迈开的第一步，直到不断发展完善它，同时反过来又导致了其它产品的开发。

年复一年，迎来了视窗系统，不是每一个人都记得在视窗系统被称作还算是一个成功的产品之前，这家公司鼎力支持这个产品的时间有多久。在我开始为微软工作时，负责热线技术支持，人们总是打电话来询问有关视窗系统的问题：有什么东西可以在它上面运行？除了随机带的那些小型的应用程序外，我仅能说出几个程序来。

视窗系统花了很久的时间才得以流行的原因有很多。一个原因是早期版本比起以后所出的版本实在是太差了，另一个原因是由于除微软以外，很少有人能够成功地开发出运行在早期的视窗系统上的应用程序，反过来说明了视窗系统对于一般的程序员来说，是众所周知的难以在它上面开发。

公平地说，这种困难不是视窗系统所独有的。视窗系统是一个以事件驱动的图形用户界面（GUI）。举个例子，苹果操作系统也是如此。诸如这样的系统，虽然对于最终用户是非常地容易，但它使程序员一下子陷入了这样一种环境：你必须获取接收到的消息，分配资

源，在输出任何东西到屏幕之前都得请求获得称作句柄的东西。简而言之，GUI 编程的困难在于你不能从一个简单的程序开始。程序员必须从一个相当繁杂的模型开始，并与一个复杂的系统相互配合。

在视窗系统下需要一种语言，就正如 Basic 语言对于 DOS 系统一样：它是供大众编程使用的。Basic 和 DOS 系统两者成功的一个因素在于 Basic 易于使用，它能够写出在 DOS 环境下运行的程序，尽管在运行时不是特别快和富有效率，但是是一个可以完成工作的程序，甚至具有工程师程度的富有经验的程序员（我记得在 80 年代初期曾为他们中的一些人工作过）也屈尊用 Basic 来开发商用程序。虽然程序有些沉闷，有时不是很漂亮，但是 Basic 使得完成项目的工期要远远快于全部用机器语言来开发所用的时间。

这或许就是为什么比尔·盖茨所谈论的要在未来开发一种假想的商用语言。他设想，这将是一种对于使用 Basic 来编程的人们来说是易于理解的语言，并且可以使用高级命令去驱动视窗程序，从而能够进行快速开发。它将是可视的，程序员们可以拥有像最终用户在视窗系统下所高兴拥有的图形工具。

直到时间过去了几年以后，Visual Basic 才渐渐由想法变为现实。

1.2 理想成真

当我第一次听到 Visual Basic 的名字时，那时我一直在编著汇编程序文档，并且正在寻找新的项目。我告诉了一个名叫 John Fine 的小伙子，他或许是微软历史上最精力充沛的项目经理（当然，除了 Steve Balmer），我还告诉了在我的第一个汇编程序产品中一起工作了几年的人们。

John 告诉我，他不久前刚接到比尔·盖茨的一项指示。微软已经获得了一项来自名为 Copper 软件公司的技术，该技术已经具有了用于视窗系统下的原型软件（原型指的是一种能迅速创建用户接口的方法）。以后几年中，我曾经看到过一些书说是 Copper 公司发明 Visual Basic ，但我认为这不是很准确的，它是比尔·盖茨的想法，即把来自 Copper 公司的图形接口软件与另一项已经存在的技术—— Basic 的线程化的 P—代码相结合，才产生了 Visual Basic 。

在这儿有必要提及另一点背景，Basic 不全是一项具有线程化的 P—代码或伪代码的技术。这项技术的产生是用来解决编译程序/解释程序这两者之间的难题，同时也是反击来自在 80 年代中期一项名叫 TurboPascal 的成功产品对 Basic 至高无上的地位发起的挑战。那时 TurboPascal 能够编译并快速地运行程序，并逐渐地拥有了一些狂热的追随者。

QuickBasic 是微软重新赢得并占据大众化编程工具市场的希望，它成功了。它提供了象 TurboPascal 那样友好的用户环境，而且它前进的更远：QuickBasic 不仅能快速地编译和运行程序，也能够使程序员停止执行程序，修改错误，然后继续执行程序——这是一种对许多程序员闻所未闻的奢侈享受。使之成为可能的部分原因在于是采用线程化的 P—代码。当程序员编辑完一条语句后,QuickBasic 就把它编译成一条中间的二进制形式，按下 RUN 键，QuickBasic 就能以比较好的性能去执行这些二进制命令，因为这些二进制命令都是线程化的，可以在不重启动的情况下将新的命令插入到程序中。

事实上，QuickBasic 已经被称为在 DOS 环境中快速开发程序的无以匹敌的工具，来自 Copper 软件公司的技术构成了在视窗系统中快速创建图形对象的工具，但在其技术之后所欠缺的是具有一种真正的可编程的语言，欠缺的是一种描述这些对象、修改这些对象、赋予这些对象复杂的行为能力的方法。回过头来很容易看到这两种技术可以相辅相成。我不知道是否有人已经将它们相结合过。但是，比尔·盖茨不仅为 Basic 写出了源码，而且他还拥有了以后通过获取技术并使其实现可视化能力的这份荣誉。

1.3 创造两者之间的结合

即使是从一个已有的了不起的界面设计工具和一个了不起的（至少是流行的）计算机语言为起点，在 Visual Basic 开发小组的前面依然有着艰苦的工作：怎样将这两部分结合起来使其象一个整体在工作呢？

开发小组通过借用面向对象模型的语法提出了解决方案。这种语法直到现在也被认为是程序设计的未来。怀疑者会反对说，Basic 太没有组织结构了，它不能支持面向对象的模型，但就其微软所考虑的，这是必然的。多年以来，面向对象的 Basic 被认为是其注定实现的目标，Visual Basic 正好是为达到这一神圣目的中的一步。现在对象恰恰是 VB 的一个方面。

面向对象语法的本质是这样的：语言是基于对象的，对象是数据的一个复合体，对象的成员包括数据域和函数。下面是一种统一表达这种关系的方法：

```
object.member
```

基于这种思想，Visual Basic 开发小组进一步提出了属性，一种特殊的数据成员。属性指的是一个对象的一般属性如：宽、高、颜色、容量等。当赋予属性一个新值后，Visual Basic 会做所有其它的事情来反映变化。如改变高度值，对象就会象魔术一样以新的大小被重新绘制。

让我们举一个纯粹想象的例子。设想你有一个名叫 Bill 的对象，它有如下属性：高度、年龄和赚钱能力，你可以使用对象语法来设置 Bill 的属性值：

```
Bill.Age = 41
Bill.Height = 6.1
Bill.EarningPower = ASTRONOMICAL
```

Bill 的属性现在被正确地设置了，你不必记任何函数名或离奇的语法，这种语法适合所有的情况。如果你想为另一个对象设置属性，即一个名叫 SteveB 的对象，你也可以这样做：

```
SteveB.Age = 44
SteveB.Height = 6.3
SteveB.EarningPower = EXCEPTIONAL
```

当然这是一个想象的例子。现在设想你有一个名叫 Readout 的文本框，你可以用这个名字去设置许多属性。例如，下面的语句可以使文本框暂时不可见：

```
Readout.Visible = False
```

另一条语句可以使它再次出现：

```
Readout.Visible = True
```

下面的语句是用来读取并且设置属性值，它将文本框的高度加倍：

```
Readout.Height = Readout.Height * 2
```

文本框的主要目的当然是用来包含文本，显然，一个特殊的属性将表示这个数据，它的名字叫 Text，因此为了把新的文本信息放入控件中，可以象这样设置属性：

```
Readout.Text = "Hello, world, it's Visual Basic !"
```

这种语法简单、优雅且富于创造性。这个系统的完美之处在于如果需要做某些工作去反映一种变化的情况（例如重新绘制控件），这些工作都在幕后完成，就像上面一样，VB 的程序员仅关心结果，属性成为了 VB 的核心，直到今天仍在起作用。

除了属性的发明（还有事件驱动模型，下一章要论述）以外，VB 另一个新颖的方面在于它给 VB 开发小组提供了一个首次也是最后一次机会去清除掉 Basic 语言本身所具有的那些难看、非常讨厌、麻烦的特性。早期，比尔·盖茨必须将 Basic 的最初版本放入内存的几千字节的空间中。想想在所受到的限制条件下，Basic 做出的许多事情真是令人惊讶。它不是一个像 Pascal、Ada 或 C++那样优雅的，具有纯粹学术气息的语言，它本来就不是。

Basic 的每一个后续版本都增添了新的特性，使得能够用它写出更好的代码。然而，向后兼容的问题一直困扰着 Basic。所谓向后兼容是指用户使用早期版本所写的代码必须继续在更新了的系统中运行。这使得要丢掉 Basic 语言中所有难看的东西是不可能的。VB 开发小组所能做的是提供一种更好的选择，并且在文档中说明旧的特性不再重要。

若可以不遵守向后兼容性的规则则确是令人高兴，这是在一种唯一的、再也不会重复出现的情况下才行。其理论根据在于当从 DOS 平台迁移到视窗系统平台后设计程序时，程序员将必须重写大量的代码（一种可靠的设想），进而必须丢弃一些关键字，因为它们违背了在视窗系统下的限制规则，其中，非视窗系统行为的主要的关键字包括 PEEK 和 POKE。

由此可知，也需要清除有问题的语法，这做一次就够了。Visual Basic 的后续版本将必须支持 VB1.0 的中的任何特性。从 DOS 平台到视窗系统平台下所作的任何改变可以一劳永逸。要清除的一个主要语法部分被称作是模块级代码（反正我是这么称呼），它是游离于任何过程之外的代码。模块级代码对于 Pascal 和 C++程序员是很烦人的，除了它遵循自己奇怪的规则以外，很象是一个主过程。通过摒弃模块级代码，Visual Basic 获得了良好的结构。

当你把 VB 与更加具有结构性的语言进行比较，诸如 Pascal 这样的纯粹的语言，你会发现一些古怪的东西。尽管如此，VB 模型中的新特性已消除了那些不是很优雅的 Basic 特性，Basic 获得了一个全新的面貌。在 VB1.0 的早期，在不告知产品名字的情况下（当时还没有正式发布），介绍给人们使用，以测试其可用性，他们都说："这看起来象 C 或 Pascal"，却从来没有人说："哦，这是 Basic"。

设计一种象那样的新语言的时候，可真是激动人心的日子。我很幸运作为一分子参与其中，并且碰到了两个具有永往直前精神的人——开发领头人 Scott Ferguson 和在 John Fine 下面工作的项目经理 Adm Rauch。他们每一个人都是伟大的天才，领导着小组中其他富有

天才能力的人们。

1.4 关于名字的由来

许多人现在或许认为 Visual Basic 这个名字像刻在石头上一样容易，但情况不是那样。在项目的开始阶段，Visual Basic 连工作标题也没有。这个项目的代码名为 Thunder(公开已经很久了)，为这项目工作的人们都相信最终 Thunder 将被一个富有鼓舞性的名字所替代，其它的软件原型设计工具都有一个象 HyperCard 和 ToolBook 那样吸引人的名字，我们认为我们的项目也会得到一个。

每一个与项目有关的人都提出了名字，但都逐个地被否决了，或者是因为没有全面地描述这项产品，或是因为该名字已经有了。(现在提出一个新软件包的名字依然是很费劲的，因为有成千上万的名字已经出现了)，最后，我们在 Visual Basic 这个名字上停了下来，这是盖茨最初的工作标题，并且这个词不仅说明了语言，也说明了整个工程项目。

对我个人而言得到了解脱，还没有其他单独的名字是用来说明这种语言和环境的。关于语言和及其环境的写作具有难以想象的困难，它们仿佛是两个分离的实体，即使仅仅是理论上的意义。最近当我将注意力重新集中于 Visual Basic 时，比以往任何时候都确信这一点：这种编程语言是如此地与图形工具紧密地结合在一起，很难将它们分割开来。这一事实也维护了 Visual Basic 定义的这一特性，即使与其他视窗系统语言如 Visual C++相比，也是如此。

1.5 定制控件免除你的担忧

定制控件经常被引用来证明是 Visual Basic 成功的关键因素之一，它可以使你象这样工作：如果你会用 C 编程，你就能创建一个新的图形对象类（控件）。你可以在很少的限制条件下有效地扩展 Visual Basic 的工作能力，也可以将你的定制控件卖给其他程序员。事实上，微软产品的市场策略总是给这个附属的市场以支持。

还是在 VB1.0 的时候，定制控件技术是在非常晚的时候被添加进来的，它对我来说几乎象是一个马后炮的想法。虽然以后得知是比尔·盖茨已经早就看到了这种技术的价值。

定制控件也反映了我涉及该项目的密切关系。尽管撰写程序员指南使我很劳累，我还是接受了一项任务，在交付日期前用不到两个月的时间将定制控件文档加进来。Scott Ferguson 给我提供了一些例子，几页注释，以及接受了我在有问题时的无数次的拜访。我需要的是一个简单但是能干一些事的例子程序，我所提出的 Circle 控件样例可能因为它的简单而受到嘲讽，但以后我发现（特别使我欣喜的是）Visual Basic 小组许多年来一直在使用它。

事实证明了程序员喜欢定制控件技术，他们喜欢扩展 VB 技术本身的这种思想：如果 VB 不能做你所需要的事情，你可以设计出你自己的控件。其中的完美之处在于新的控件能够巧妙地适合已经存在的编程模型。例如，控件的使用者可以获取和设置属性值，就象是设置标准控件一样。