用C语言开发图形软件

H

中国科学院希望高级电脑公司

一九八八年四月

前　言

几年前，人们还认为计算机图形学是没有必要、缺少实用价值、而且费用过高。但如今，硬件和软件具有的速度和能力，使图形学不仅使用方便而且是许多应用的基础。台式排版机的出现使现代的计算机能把图形和正文结合在一起。不久，数据库会象存贮正文信息那样存贮图象。

计算机应用的范围不断扩展。过去的一般时间内，计算机主要用于计算，但现在却扩展到所有的领域，协助人们完成各种各样的工作。借助图形学计算机可以显示脑电图、天气形势图以及地质构造等等。图形系统还可用于图形艺术、医学、科学、机器人学、安全保密和质量控制等领域。至于象绘制草图、生产计划和设计等应用已是相当普及的了。

编著这本书的目的是为读者建立起如何使用C语言开发图形系统的总概念。为此，本书专门编写了一个实际图形系统GRAPHIQ，其源程序打印在附录A中。它的每个函数在本书的各个章节中给予详细的论述。在图形系统GRAPHIQ中，还提供了一个工具函数包，帮助用户对IBM的增强型图形适配器（Enhanced Graphics Adapter—EGA）和AT&T的图象处理板（Image Capture Board—ICB）进行编程。对于象C语言这样的开放式程序设计语言，这类图形工具包具有特殊的价值，它扩充了程序设计语言描述图形的能力。

本书还介绍许多相关的技术，以帮助读者理解图形软件开发这一复杂的领域，详尽地叙述了用于图形学程序设计的算法、图形编辑、硬拷贝的创建、用户存取方式、通讯，以及高效的文件系统的设计。这些功能都在GRAPHIQ系统的函数中有所对应。

本书除了详尽介绍各种图形处理函数外，还提供了许多用于复杂图形程序设计和开发所需要的有用的工具。这些工具包括支持串行和并行输入／输出的通用函数、上推式菜单、以及图形正文。一个完整的图形正文字形的源代码也列于本书中。《高级语言C的图形系统开发》也提供许多外部设备的接口技术，例如打印机、绘图仪、数学化仪和鼠标器等等。

本书将慷慨地、开放地提供GRAPHIQ的全部源程序和读者共享这些图形处理软件。本书作者希望能通过从图形程序最核心部份开始，一步一步地向外地介绍整个程序的源代码，让读者了解图形程序设计的秘密所在。

（1）你对C语言掌握程度如何？

如果你刚刚开始学习用C语言编程，那么会感到本书的某些章节难以理解。本书不是讲述有关C语言程序设计基础的书籍，而是对中高级C语言程序员讲述怎样进行图形程序设计。

然而，如果你正在开始学习C语言程序设计，那么本书对你也是有用的。通过研究用C语言编写的程序系统GRAPHIQ，你可以学到许多有关C语言的知识。随着对C语言知识的增加，就可以理解本书所给出的函数是怎样构造出来的，也可以学会在编写图形程序时怎样改进这些函数。

（2）怎样使用本书

本书是以二个渠道方式向读者传授知识的。读者可以阅读每一章中的详细解释，也可以阅读附录中的源程序代码。

前两章从风格和概念的角度讲述图形程序设计的原理。你可以在这两章中找到一些有关C语言程序设计和设计图形软件的通用规则。

第三章介绍GRAPHIQ程序本身。这一章解释了在GRAPHIQ系统背后的目的，并叙述了该系统总的特征。

第四章和第五章讲述了用于图形程序设计的算法。你可以找到改变象素颜色、画直线、画图、填充区域等算法，以及许多其它的常用于图形程序设计的功能函数。

第六章和第七章讨论了图形编辑（在此要完成绘图工作的大部份）和文本的生成。第七章是讲述怎样生成和存贮图形正文字形。

第八章讲述怎样使用图形硬件和软件生成硬拷贝。你可以看到怎样将图形信息传送到打印机和绘图仪的过程。

第九章和第十章叙述的是计算机系统与用户的通信方式。讲细地介绍了上推菜单和定位输入器的使用情况。你将要学会怎样快速地将图形和正文文本从显示屏幕上快速地移出或移入。

第十一章和第十二章讲述计算机硬件与外界通信的方式。本书提供了有关串行接口和并行接口的详细资料。读者可以从中了解图形存放到软盘或硬盘上的方法，以及创建命令文件並以远端控制的方式运行你的系统。

第十三章讨论了文档这个十分重要但却常常被忽视的问题。本章还讲述了怎样把图例和命令作为通信的工具。

第十四章和第十五章讲述Micro—soft C编译程序和链接程序的使用方法，同时讲述了编译和链接的一般技术。同时还指出了GRAPHIQ编译和链接的特殊要求。

在附录我们给出了GRAPHIQ系统的全部源程序代码，其中还包括有该系统的命令汇总和有关使用C语言及汇编进行程序设计的要点。同时在附录中，我们还讨论了一个崭新的领域——视频程序设计，並提供了几个有用的程序。用户使用这些程序可在ICB与EGA之间传送图象。

如果你的PC机使用的不是EGA，那么对于本书有关硬件的函数要稍做修改才能使用。由于本书所提供的工具包都是源代码，所以很容易进行必要的修改。

在图形学领域，每天都有新的标准出现，以致谈不上是什么标准。因为当前没有标准的硬件和软件可以使用，所以不可能提供和维护在所有设备上都完全兼容的软件。

专为本书开发的工具包软件和GRAPHIQ的源程序，已存入软盘中，並可向读者提供，该盘中还包括GRAPHIQ的可执行文件和一个目标模块的函数库。

目　　录

第一章　图形程序设计 ……（ 1 ）
1.1　成功的图形程序设计的原则 ……（ 1 ）
1.2　自顶向下设计方法 ……（ 2 ）
1.3　自顶向下程序设计的某些误解 ……（ 2 ）
1.4　图形程序设计的忠告 ……（ 3 ）
1.5　图形编程的风格 ……（ 5 ）
第二章　图形软件设计 ……（ 7 ）
2.1　显示模式和显示页码的设置 ……（ 7 ）
2.2　在EGA上增强的图形模式 ……（ 11 ）
2.3　使用DEBUG画直线 ……（ 12 ）
2.4　用C语言进行图形程序设计 ……（ 14 ）
2.5　C语言图形程序的构成 ……（ 16 ）
2.6　结构流程图 ……（ 20 ）
2.7　内存使用模型 ……（ 21 ）
2.8　其它各种显示适配器 ……（ 22 ）
第三章　GRAPHIQ：一个示范性的图形原型系统 ……（ 24 ）
3.1　GRAPHIQ程序概况 ……（ 24 ）
3.2　GRAPHIQ支持的设备 ……（ 24 ）
3.3　数据库设计 ……（ 25 ）
3.4　GRAPHIQ的文件结构 ……（ 26 ）
3.5　GRAPHIQ的局限 ……（ 30 ）
3.6　改进GRAPHIQ的建议 ……（ 31 ）
第四章　图形学算法 ……（ 32 ）
4.1　坐标系统 ……（ 32 ）
4.2　绘图元素 ……（ 34 ）
4.3　直线算法 ……（ 35 ）
4.4　画圆算法 ……（ 39 ）
4.5　绘制弧线 ……（ 42 ）
4.6　填充算法 ……（ 43 ）
4.7　绘制交叉阴影线 ……（ 47 ）
4.8　抖动 ……（ 48 ）
4.9　裁剪 ……（ 58 ）
4.10 字形表 ……（ 61 ）

4.11 从算法到C代码 …… (64)
第五章 图形变换 …… (65)
5.1 对象 …… (65)
5.2 旋转 …… (66)
5.3 平移 …… (69)
5.4 比例变换 …… (69)
5.5 组合变换 …… (70)
5.6 投影 …… (72)
5.7 图形变换函数程序设计的注意事项 …… (74)
第六章 图形编辑 …… (76)
6.1 图形编辑的功能 …… (76)
6.2 各形编辑中的菜单和命令 …… (76)
6.3 画一个点 …… (79)
6.4 画直线 …… (85)
6.5 画圆周和圆弧 …… (94)
6.6 画矩形 …… (103)
6.7 填充矩形 …… (104)
6.8 复杂的填充 …… (106)
6.9 直线、圆、矩形和填充的组合 …… (109)
6.10 画笔的使用 …… (111)
6.11 图形的拷贝 …… (112)
6.12 编辑功能的结合 …… (115)
第七章 正文绘制 …… (116)
7.1 ASCII字符集 …… (116)
7.2 笔划字形库 …… (120)
第八章 打印和绘制图形 …… (138)
8.1 打印图形 …… (139)
8.2 绘制图形 …… (148)
8.3 打印和绘制图形函数的协调 …… (160)
第九章 菜单设计 …… (162)
9.1 菜单设计 …… (162)
9.2 菜单项的选取 …… (170)
9.3 加速菜单显示技术 …… (176)
9.4 键盘交互 …… (182)
第十章 定位光标 …… (189)
10.1 定位光标的设计 …… (190)
10.2 定位光标的编码 …… (191)
第十一章 并行和串行接口技术 …… (204)
11.1 并行端口 …… (204)

11.2 串行端口……（210）
第十二章 维护方式……（235）
12.1 图形系统中的函数……（235）
12.2 程序的启动与结束……（238）
12.3 用于图形和命令的文件……（248）
第十三章 图形文档资料……（254）
13.1 图形任务的甄别……（254）
13.2 自用的图形程序……（256）
13.3 专为用户设计的图形程序……（257）
13.4 用于市场销售的图形程序……（257）
13.5 图形肖像和文字……（258）
第十四章 C语言编译程序的使用……（261）
14.1 Microsoft C语言编译程序……（261）
14.2 图形工具包……（265）
第十五章 链接图形工具包……（273）
15.1 程序库的建立……（273）
15.2 链接程序的使用……（281）
附录A GRAPHIQ源程序清单……（284）
A—1 怎样使用GRAPHIQ源程序代码……（284）
A—2 GRAPHIQ源程序代码……（285）
A—3 GRAPHIQ的头文件……（391）
附录B GRAPHIQ的命令语法……（424）
B—1 总功能描述……（424）
附录C 使用汇编程序进行优化……（428）
附录D 使命令变成可访问的……（438）
D—1 AUTOEXEC·BAT文件……（438）
附录E 电视图形……（438）
E—1 摄像机的使用……（439）
E—2 传送算法……（439）
E—3 光栅图形输出到磁盘……（446）
E—4 从磁盘输入到光栅……（448）
E—5 TRANSFER·C程序……（450）
E—6 其它参考资料……（454）

第一章　图形程序设计

在程序设计中，有人靠直觉，有人则按步就班地进行。对于一个程序设计项目，最好是在一个完好构思结构的控制下充分发挥程序员的直觉。一般说来，图形处理程序比非图形学程序复杂得多。由于这个原因，图形程序开发难度比较大，也不太好标准化，使用起来也比较困难。但是，只要一开始在头脑中，就建立起清晰的概念，便可以克服图形软件开发中可能出现的混乱现象。

在本书中，读者可以找到使用C语言进行图形程序设计的许多细节。由于本书是面向中高级的C语言程序员而编写的，一般读者所熟知的许多术语，本书没有重新进行定义或作进一步的介释。除了阐明一些中高级的图形学概念以外，本书也没有讲授C语言的基础内容。如果读者想要弄清“函数”、“偏移量”或“变量”等的定义是什么，则可参考有关C语言程序设计方面的教科书。

1.1　成功的图形程序设计的原则

成功的图形程序设计原则与其它成功的程序设计的原则並没有什么不同。处理纯文本的系统可充分利用了C语言（和其它程序设计语言一样）处理正文方式见长这一事实。但在图形程序设计中，人们将要使用正文形式的代码绘制点、直线，以及其它非正文的图形构造。这可在一些读者中会引起某些混乱，所以需要经常强调代码的结构，即使对于输出仅仅局限于文本信息的程序也需如此。

（1）坚持到底

成功的程序设计的第一个原则是要坚持。如果你对所做的工作有兴趣，那么坚持下去就不会是一件困难的事情。图形学问题不象仅涉及正文的那些问题那某明显。图形程序设计需要较长和较复杂的调试过程。

（2）结构化程序设计

成功的程序设计的第二个原则是使用结构化设计的能力。计算机程序从一般到特殊地进行构造。象建造一幢大楼那样从简单的概念开始，逐步深入直到所有的细节，最后得到以混凝土、木材和钢材等材料构成的结构。

当然，建造一幢大楼可以有各种各样的方式。你可以在院子里堆放一堆木头，用钉子将木板钉在一起，造成一幢木制小楼。这是一种自底向上的建筑方式，也许有人希望用这种方式搞建设。因为没有考虑底层部件对上层部件的支撑，所以这种建筑形式是不安全的。换句话说，这种方法仅仅是在建造而没有设计过程。

设计是通过运用规划将某种设想加以实施的过程。这一概念在德文中是“Das Ein”，意指“在其中”的含义，给出了设计的另一种定义。这两者之间仅有着很微小的差别。在设计的时候，你要问自己所要设计的东西应该是怎样的。

（3）分而治之

第三个成功的程序设计原则是“分而治之”。在十四世纪有一位名叫威廉·奥克汉姆的经院哲学家，在试图将铅炼成金子的过程中形成了一种原则，后来人们称之为“奥

克姆瓜分法”。按照这种方法，任何问题均可分解为若干部分，並独立地逐个解决问题的各个部分，最终便可解决原来的整个的问题。C语言的程序设计也是基于这种原理来设计图形系统的。在C语言中，你可以或多或少地将程序分解为若干个独立的函数，然后对这些函分别地进行设计和调试。这种方式是实现自顶向下程序设计方法的基础。

1.2 自顶向下设计方法

本书中的程序系统是按照自顶向下的方法实现的。自顶向下的设计方法之所以能在图形程序设计中发挥巨大的作用，是因为这种方法能帮助设计者组织那些非常复杂而又难以用词语表达的概念。

经验表明，如同建造房屋的过程那样，与其他方法相比，自顶向下的方法设计程序系统成功率大得多，也较为安全可靠。自顶向下的程序设计方法是要帮助程序是考虑软件总体的设计目标，而不是将混杂在一起的函数装配成实现某些功能的手段。

（1）对问题的描述

自顶向下的设计方法是以描述准备解决的问题为始点。例如，某一个图形处理问题可能需要查看指定的月份中库存的最高数量、最低数量和单价。价格数据存放在磁盘中，需要将这些数据转换成为直棍图，以价格为纵座标，以时间为横座标。

在这个实例中，你可以看出，在需要问题描述中，包括了程序所需要的输入和输间出。输入是从文件中读取的库存货物的价格和日期，输出是显示某段时间内价格随出变次而变化的图形表示。

开头，你无需知道要在什么设备上输出。输出设备可能是一个计算机显示器、一张纸、幻灯片或其它一切图形输出设备。在刚刚设计时，不必要知道最终使用什么样的设备，关键的是在输入和输出之间传送的应是那些东西。

（2）弄清楚所要达到的目标

如果没有弄明白所要达到的目标是什么，就不要开始进行设计。一个设计得比较好的程序就象一个洋葱头一样，在其最外层，你可以见到问题的全貌。当你剥开靠处的一层，便可见到靠里的一层，在必要时候，还可回到最外面的那层。为了解决最外层的问题，必须清楚地定义所要达到的目标才行。

对问题的描述在于对该问题的理解。在你的日常生活中，这点可以得到验证。为了要解决一个问题，你必须集中精力对问题每个方面进行准确的理解。为了做到这一点，你必须始终记住你所要达到的目标。如果把所要求的目标始终贯穿于整个设计过程之中，便可以使你的程序达到恰到好处的简洁。

1.3 自顶向下程序设计的某些误解

这些年来，存在着一些对自顶向下的结构化程序设计的误解。这里讨论了其中最普遍的两种。

（1）从开始到结束

一种对于自顶向下设计方法的误解是认为程序员不应该在设计过程的早期关心最终的产品，最终产品可以从设计过程中自然而然的逐步进化而得到。

正确的自顶向下实现方法是认为程序系统应由嵌套着的不同层次的函数构成。最外

层函数必须包括开始和结尾部分。系统使用的细节要逐层地加到系统外壳上。自顶向下的设计过程是指这种层次深度的顶和底，而不是程序的开始和结束。

（2）永远是自顶向下

另一种对自顶向下程序设计的误解是想象每一件事情都要自顶向下地完成。实际上，应用系统的设计是自顶向下完成的，但其构造（即编码）却是自底向上完成的。

在设计房屋结构时，总是象结构工程师所做的那样，进行自顶向下的设计。显然，低层支撑着高层。屋顶的重量要加到地板上，而屋顶和地板一起的重量要由地下建筑来支撑。但在实际盖这幢房子的时候，果真先建造屋顶，那在大多数情况下将是很破费的。类似地，应用程序的实际编码过程也是先建造低层的输入输出程序。这些程序必须放在主程序的适当位置，才能开发成功的图形系统。

1.4 图形程序设计的忠告

在使用本书和本书提供的图形函数时，你可能希望考虑一些近年来证明是行之有效的一些技术。每个程序员都有着自己的偏爱和编制程序的绝招。以下是一些建立软件系统有益的忠告。

（1）应注意两种不同的数域

在图形学中，“域”或“世界”这一概念是十分重要的，它有着特殊的含义。在人们通常使用“外部世界”或“世界观”中，使用“世界”这个词表示客观的存在和状态等。在本书中，我们要涉及到许多域，或说许多种世界。例如，在使用实数时，我们说程序是在实数域或“实数世界”中。通过使用世界或域这一概念，在图形程序的不同部分人们可将其适用的原则牢记在心中。

记住两种类型的数世界是非常有用的。一种是“数世界”，另一种是“偏移量世界”。在偏移量世界中的数值永远是一个小于数世界中某个数值的整数！当你要定义数项变量时，需要使用数世界；而当你试图在一个文件中或字符串中定位时，就使用偏移量世界。对你要现程序中使用的每个数，最好自问一下：“这是一个数呢？还是一个偏移量？”。这将是十分有益。

例如，显示宽度可能是640个象素。显示宽度是一个不变的数而不是一个偏移量。你要用到的偏移量是从0到639。这一点看起来十分明显，但如果你没有明确地分开这两种数的世界，就很容易在图形程序设计的过程中发生混淆。

（2）直接地操纵内存

基本上有两种不同的方式，可以将象素（光点）显示到显示输出设备上。第一种是通过BIOS（基本的输入／输出系统）；第二种是通过直接地修改图形区内存。图形程序系统需要尽可能快速地显位图形。

因为通过BIOS显示图形需要一些额外的步骤（产生一个中断，并由BIOS确定系统当前使用的适配器的类型），所以除非在无法预知使用适配器的类型的情况下，一般要避免使用BIOS。达到较高速度的较好的方法是直接地对显示RAM（动态存贮器）进行存取。

对于本书多数的读者，已经有个一些C语言程序设计的经验，所以我们认为，多数读者已经知道了RAM是什么，BIOS是什么，以及怎样生成一个中断。如果有些读者对

于这些问题不甚了解，请在阅读本书以下各部分之前，先熟悉这些术语。可以读一下目前已有的非常出色的C语言基础、汇编语言和计算机体系结构方面的教科书。

（3）使用汇编程序优化C语言代码

我们主张，在汇编程序级上使用一些技术对C语言代码进行优化，以便提高图形系统运行的速度。首先将C语言编译成汇编语言的程序，并仔细阅读这个程序，寻找能以较高的速度完成同样任务的方法。当然，这是很花费时间的事情，但这将意味着将一个较慢运行的图形程序升服成为一个运行很快的图形程序！在图形程序系统中某些基本函数结合要重复使用千万次。如果这样的函数没有进行优化处理、累计的效果将会大大地降低系统的运行效率。应集中全力对如Pointc（ ）和象素块搬家那样的基本函数进行优化处理。

尽管这是一本有关使用C语言进行图形程序设计的书籍，但读者会发现如果能掌握机器的指令系统，就会成为更为出色的程序员。在附录C中，我们为高级读者准备了使用8086汇编程序进行优化C代码的内容。

（4）兼容能力的考虑

当第一批IBM个人计算机被推出之后，许多厂家和公司都推出了自己的兼容机。今天，其中的许多产品都被挤出了市场。只有那些最初就决定与IBM机器100%兼容的个人计算机才生存下来。

当你设计图形系统时，如果没有仔细地选择图形显示设备，就会遇到许多有关兼容性的问题。为那些没有人使用的硬件开发软件是毫无意义的。

（5）把程序看作命令

你的软件可能要采取菜单形式，但不要忽视生成可在DOS命令级上执行的图形命令。DOS和UNIX操作系统允许使向管道和重新定向一类的操作，以便以复杂的方式将命令链接在一起。切实把图形程序与操作系统相分离，使用操作系统环境可使系统变量生效。如果你还不熟悉这些概念，请阅读DOS手册中有关管道、标准输入／输出的重新定向和环境等若干项的内容。

（6）使程序尽可能简单

图形程序系统的发展趋势图形程序运行越来越快，即使是使非常复杂的程序也是这样。读者应该集中力量学习彩色的使用，上推式菜单设计和其它软件技术。

（7）使用快速写屏幕函数Screen—Write（ ）

使用Sprintf（ ）函数，而不使用Printf（ ）函数，并把一字符指针传送给写字符到屏幕的函数。这样可以控制颜色和速度。例如，通常情况下，你可能会用如下的语句，生成一命令提示：

Printf（“Default filename：%S”，filename）；不要使用这种工作方式，最好改用下列语句：

Sprintf（buffer，“Default filename：%S”，filename）；

Write—Screen（buffer，X，Y，RED，BLUE）；采用这种方法，你可以控制提示的显示位置，还能控制前景和背景的颜色。当你想要显示一个简单的字符串时，可以省略掉Sprintf（ ）并将字符串指针直接传送给Write—Screen（ ）函数。Write—Screen（ ）是一个快速写屏幕的函数。由于Printf（ ），Sprint（ ），

和fPrint( )，和fprintf( )需要大量的系统开销，因而降低系统处理速度。

（8）使用标准化的设备座标

每人使用显示设备都有其自身的宽度和高度。如果读者希望在图形应用系统中使用数种。图形设备（例如，绘图仪和CRT显示器一起使用等情况），就需要使用NDC(标准化设备座标）系统。

在程序中采用NDC是与其驱动程序相分离的。当你把一个点的座标传送到设备上去的时候，这些座标应当在NDC单元中。当设备接收到座标时，由驱动程序将其转化成为实际设备的真正座标。

例如，你所用的座标系统可能从左下角的（0，0）点开始，沿X轴和Y轴扩展到（32，767）。所有在系统中进行的操作，都必须在同一的，从0到32，767个NDC单位的范围内进行。当被选择的设备接收到以这种单位表示的座标时，设备将要将其转换成为自身显示所需要的数值。例如，一部EGA（增强的图形适配器）设备将要把32，767单位的数域转换成640单位的水平数域。垂直方向上，这种设备可将32，767单位的数域转换成200单元的数域。

当然，为了保持圆的形状，而不使其变成为椭圆，保持正方形而不使其变成长方形，这种设备必须在NDC上乘以一个比例的变换因子。这样，每个设备的驱动程序都包括将1：1的NDC单位的比率转换成所使用显示设备所需要的比率。

（9）固件

尽可能使用显示适配器，绘图仪和打印机板上提供的固件。固件可以帮助你有效地编用各种设备。一个设计精良的设备，都共享些软件固化在系统中的固件中。因为固件使码是在ROM（只读存贮器）中，所以其运行速度要比以任何基于RAM的代码来得高。你应尽可能使用这些已存在的实用程序，而不必花费时间去重新编写它们。

1.5 图形编程的风格

本书在编程时采用了特定的风格。虽然各个有其自己独特的风格，但如果你想了解一些本书中所采用的编程风格，最好还是读一下包含在本书中的一些程序。

（1）程序外观与其阅读的清晰性

显示在屏幕上的程序代码的编写方式与其能否在今后被人轻松地读懂有很大的关系即使是现在编写的程序，很可能以后要回过头来对它进行修改。因此要保持良好的编程风格，改善程序代码的外观和提高其阅读的清晰程度利用。C语言可以编写外观精美，易于阅读的程序，但有时能生成难以阅读的程序代码。作为一个极难阅读的程序，请看Functin 1—1——“Beauty and the Beast”。

Function 1—1中的两个例子，表示的是同一件事情，但很明显能够看出哪个代码可被人读懂。虽然这是两种极端的情况，但是两个函数却能表明程序的结构和编程一风格是与程序的可读性密切相关的。变量名应清楚地表明每个变量的用途。“BEAST”根本上就没有什么风格，但却遵守了C语言的语法规则，也能调试运行成功。由此可见程序代码的外观是很重要的。

在这些，还需要扩展对于图形概念的理解，不能仅仅局限于直观的理解。你在编码中所表现出的风格本身就构成了一种图形。一般而言，包括了亮区和暗区中的各个方

FUNCTION 1-1

Beauty and the Beast

```
BEAUTY:

main()
{
unsigned int albatross, marlin, whale;
char sea_string[81];
double stars_in_sky, fish_in_sea;

while(stars_in_sky > fish_in_sea)
     {
     save_whales(&albatross, &marlin, &whale);
     which_greater(&stars_in_sky, &fish_in_sea,
                   sea_string);
     }

printf("%s", sea_string);

}


THE BEAST:

main() { unsigned int albatross, marlin, whale;
char sea_string[81]; double stars_in_sky, fish_in_sea;
while(stars_in_sky > fish_in_sea) { save_whales(&albatross,
&marlin, &whale); which_greater(&stars_in_sky, &fish_in_sea,
sea_string); } printf("%s", sea_string); }
```

面，人们所谈论的是直线和直线簇，彩色，区域，以及表格等。C语言程序的文本是特定方式书写的字符表达式和逻辑表达式，但它同时也是一组字符的集合。这组字符的编写也是一种图形构成的过程。

（2）空白符

程序源代码由字符、换行符号和制表符等组成。应当在程序中恰当地使用空白符，特别是用在同一文件的各个函数之间。空白符实质上是构成程序外观的一种方式，而且行首空白的使用可以清楚地表明逻辑控制的层次。

（3）变量名字

在变量命名时应使它的名字能够清楚地表示它的含义。在变量的用法发生改变时，应当相应地改动变量的名字。

首要的是千万不要懒惰！要发展自己的风格并坚持使用它。实际上，并没有哪一种风格可被举世人人所接受。一些程序员喜欢用制表键写出行首的空白并从此处顺延下去。另外一些人喜欢将左括号写在初始变量的左边而不是下边。只要你的程序能够被那些需要读你程序的人读懂，那么这种风格就是合适的。

第二章 图形软件设计

图形学的结构化程序设计要求能够灵活地使用内存存贮图形信息。当你用计算机画直线或其它图形时，实际上就是给计算机的内存空间注入不同的内容。这比起把图形画在纸上来，既有优点又有缺点。用计算机作图的技术决不是显而易见的，它要受到无情无义的约束，但如果你掌握了这种技术，便可以以比手工方法快得多的速度得数据转化成有意义的方块图形。你将会发现，图形程序设计与普通的绘图有很大的差别。

这一章将介绍对图形内存空间的管理。这是图形程序设计中最为重要的一部分。在这一章图形程序设计中，你将单会使用计算机内存作为输出设备。正如将要向你介绍的那样，可以通过改变内存的内容，便可改变显示屏上光点的颜色。

虽然图形程序也使用打印机和绘图仪等外部设备绘制图形单元，但通常程序都是将代表图形的内容写到显示缓冲区或从内存的某一部分中读出。这不是对内存仅有的操作方式。还有许多对显示的缓冲区读出或写入进行管理的隐蔽结构。

2.1 显示模式和显示页码的设置

在学习显示内存内容处理之前，必须能够设置显示方式才行。增强的图形适配器（EGA）能够支持几种类型的显示器，包括简单的图形显示器，IBM的增强型彩色显示器，IBM的单色显示器，以及其他许多类似的设备。它是一种相当通用的产品。

EGA适配器使用几种不同的模式控制其整个运行过程，这和使用的硬件以及各个开关的设置有关。对于“模式”（mode）一词，无论在本书或在计算机通用的术语中，经常产发生一些歧义。只要把模式、全程量和系统状态这几个术语作为系统（无论是软件还是固件）可以识别的定义信息的整体，在理解模式的含义上就不会有什么困难。

为了能使EGA支持更多的设备，增加了四种全新的模式，原来在IBM的彩色／图形适配器（CGA）上使用的方式仍然有效。适配器CGA支持的点从模式0到模式7。四种新增加的模式是模式13到模式16，能够使图形增强到16种颜色（从64种色的调色板中选择），640×350的高解像度显示，以及单色显示器上的图形显示。

表2—1中列出了EGA可以采用的各种模式。注意，由于EGA不使用模式8、模式6和模式10，所以在表中没有列出。这三种模式只在IBM PCjr 机上才使用。如果你使用PCjr机器，那么本书中所给出的函数仍然有用，只要按照IBM PCjr的文档资料稍作一些修改即可。模式11和模式12用以加载文体区域。所有其它的从0到16的各种模式对应所有应用都是有效的。当还要看是否在系统中插入了必要硬件。

如果开关设置得不对头，一些模式就无法使用。因此，要十分注意EGA安装指南上的有关内容，并按它的指示适当地设置适配器后部的四个开关。

在本书中，我们选用了模式14。这种工作模式给出了16种固定的颜色和640×200象素的分辨率。如果你希望支持其它模式，那么改变几个有关的常数是没有什么困难的。我们已经将一些常数（如显示的宽度、高度和模式等）集中放在一切，以便能比较容易地改变这些设置。由于用来讲述支持其他模方式要花费约较多的时间，可能会冲淡本书中图形程序设计概念的重要性，所以将按照使用说明改变模式的工作留给读者自己完成。

EGA Mode	Text or Graphic	Display connected (Color, Enhanced or Mono)	Pages	Colors	Resolution (horiz x vert)	Characters (horiz x vert)	Cell (horiz x vert)	
0	T	C	8	16	320 x 200	40 x 25	8 x 8	64K
1	T	E	8	16/64	320 x 350	40 x 25	8 x 14	64K
2	T	C	8	16	640 x 200	80 x 25	8 x 8	64K
3	T	E	8	16/64	640 x 350	80 x 25	8 x 14	64K
4,5	G	C,E	1	4	320 x 200	40 x 25	8 x 8	64K
6	G	C,E	1	2	640 x 200	80 x 25	8 x 14	64K
7	T	M	8	4	720 x 350	80 x 25	9 x 14	64K
13	G	C,E	2 4 8	16	320 x 200	40 x 25	8 x 8	64K 128K 256K
14	G	C,E	1 2 4	16	640 x 200	80 x 25	8 x 8	64K 128K 256K
15	G	M	1 2	4	640 x 350	80 x 25	8 x 14	64K 128K
16	G	E	1/2 1 2	4/64 16/64 16/64	640 x 350	80 x 25	8 x 14	64K 128K 256K

Sources for data: IBM Technical Reference, PC Magazine

TABLE 2-1. Modes of Operation for the EGA

表2—1　EGA的操作模式

同样地，对EGA细节本节将不作更多的讨论，读者可参阅其它的有关书籍。为了避免冲淡本书复杂的图形概念这一重点的内容，我们仅仅把EGA作为一种方便的和有效的示范性工具。

（1）兼容性

如果在你的图形系统中兼容性是一个关键的问题，那么最好使用本书提供的与EGA　BIOS有关的几个函数。在EGA中所包含的BIOS，虽然效率较低但却是通用程度较高。这些程序兼自动地补偿了业已设置了的模式。但是，如果象大多数用户希望的那样的若想使程序运行快一些，则要使用那些能通过显示控制器直接控制硬件的函数。

为使你的图形函数能够最大限度地兼容各种图形设备，应该把如下四点当作风格和结构记在心中：

1.将显示缓冲区的起始地址放在一个全局变量中。

2.必须了解在显示设备上起始地址的物理位置。例如，在EGA上是左上角，而在ICB上则是左下角等。

3.显示器的分辨率将要确定显示象素的数量，因此也要确定对应于显示屏上水平距离和垂直距离显示记录的数量。

4.对每个象素在彩色记录中位的格式将要决定彩色图形的结构。

（2）显示模式的设置

你可以写出一个简单的程序，来改变DOS命令行的模式。DOS MODE命令可以

完成这项任务，但它局限于由CGA和单色显示适配器（MDA）所支持的那几种模式上。作为图形程序设计的第一个练习，最好将在Function 2—1所示的程序MODEGA·C输入到计算机中，并将其编译和执行。

程序MODEGA·C利用在EGA板上ROM片子中的EGABIOS，可以设置由EGA支持的所有模式。EGA模式是一个事先已定义好了的状态，设置在EGA的存贮器中并控制着EGA的全部功能。

FUNCTION 2-1

MODEGA.C

```
/* MODEGA.C sets ega mode */

#include <stdio.h>
#include <dos.h>

union REGS regs;

main(argc, argv)
int argc;
char *argv[];
{

if (argc == 2)
     modeset(atoi(argv[1]));

if (argc != 2)
     printf("EGA Mode is %d", modeget());
}


modeset(mode)
int mode;
{

regs.h.al = mode;
regs.h.ah = 0;
int86(0x10, &regs, &regs);
}


modeget()
{

regs.h.al = 0;
regs.h.ah = 0x0f;
int86(0x10, &regs, &regs);
return(regs.h.al);
}
```

MODEGA函数读入所希望的模式的数值。它是命令行的第一个变量。例如，可以通过下列命令设置模式6：

E：>modega 6

当你键入这条命令后，系统立即设置了你所希望的那种模式。如果设置成功，你会看到屏幕将改变成你所希望的那种式样；如果设置不成功，该命令就什么作用也没有。本程序没有提值出错信息，这是必要的，因为这样可以执行DOS命令的软件中调用这些命令。如果已经将参数传递给了父进程，就不必再在屏幕上打印信息，以免打乱父进程的操作。

如果你输入了一个不带参数的命令，那么MODEGA就会显示出在输入该命令时系

统所使用的模式。例如，按下列格式输入：

E：＞mobega

如果当前的模式是6，则会导致下列的系统输出：

E：＞modega

EGA Mode is 6

E：＞-

在MODEGA·C程序中还包括一个称为modeset( )的函数。这个函数是使用EGA的工具包中的第一个函数。如果仔细研究一下这个函数的源代码，你就会立刻明白它的工作原理。

（3）显示页码的设置

■ FUNCTION 2-2

PAGEGA.C

```
/* PAGEGA.C sets ega page */

#include <stdio.h>
#include <dos.h>

union REGS regs;


main(argc, argv)
int argc;
char *argv[];
{
int page;

if (argc == 2)
     page = pageset(atoi(argv[1]));

if (argc != 2 || page == -1)
     printf("EGA Page is %d", pageget());
}


pageset(page)
int page;
{

if (page < 0 || page > 7) return(-1);

regs.h.al = page;
regs.h.ah = 5;
int86(0x10, &regs, &regs);

return(page);
}


pageget()

{

regs.h.al = 0;
regs.h.ah = 0x0f;
int86(0x10, &regs, &regs);
return(regs.h.bh);
}
```

第二个非常有用的程序是PAGEGA·C，其源代码列在Function 2—2中。PAGEGA使你能在EGA上选译要显示页。页是RAM区域的一部分，对某一种给定的模式，建立了该模式所需要的显示缓冲区。有效的显示页数既依赖于适配器上的RAM空间，也依赖于所选译的模式。由于每一种模式都有不同的要求，所以显示缓冲区的大小也不一样。并且，整个被分配的RAM空间也相应地可以分成若干页。对于EGA上给定数量的RAM在每种模式下有效内存的页数，请看表2—1。

可以按照运行MODEGA同样的方法，通过运行PAGEGA来选择显示页码（0—7）。只有一点与运行MODEGA不同，这时在命令行输入的参数是显示页号，而不是所希望的模式。这时所选译页的内容将立即显示在显示器上。可以采用快速页交换方式的优点，以加快图形应用程序运行的速度。

2.2 在EGA上增强的图形模式

我们之所以从EGA上的模式6开始进门图形程序设计，这是因为它是所有模式中最简单的一种。只有两种可以使用的颜色，即BLACK（0）和WHITE（1）。显示缓冲区内存的首地址为B800（十六进制）段，在偏移了（640×200）／8—1之后，便是结束地址，即结束地址是B800：3E7F。象素到内存每一位的映象是直接了当的。显示缓冲内存的每个字节刚好对应着8个象素，每一位对应着一个象素。这些内存中的位和象素还对应着显示屏幕上的一个点。图2.1中表示的是在EGA显示缓冲区中一个典型字节的位映象情况。模式6是图形显示中比较简单的一种模式。

在这里，模式6的选用仅仅是起到一种示范解释作用。EGA能够做到的远不止这些，它可以采用高分辨率图形工作模式，并可从64种色彩的调色板上最多选用16颜色的能力。用于画直线的那些

位的位置：	7 6 5 4 3 2 1 0
从左到右的象素映象：	1 1 1 1 1 1 1 1
对应的16进制数：	F F

图2—1 对EGA模式6的一个字节位映象

普遍性原则与用于画其它图形的原则是一样的。但是，CGA采用图形控制器，与EGA所采用直接的方式的图形控制器是不同的。

（1）EGA图形控制器

图形控制器是由IBM公司专门设计的芯片，用作程序员和显示RAM区的中间媒介。人们不能直接修改RAM，而是必须首先向图形控制器中的索引寄存器（端口0×3CE）发送一个索引码。这个索引码调用了控制器上的一种特指功能。你可以在本书中叙述的函数里找到这一过程的实施情况，如果你希望了解显示控制器工作过程的细节，那么通过研究画点和画线的函数，一切便可知晓。

如果你将EGA的模式改为14，并试图不使用显示控制器而改变显示内存区的内容，那么你会发现这样做是行不通的。显示RAM对你来说不是直接所能看到的。在EGA的彩色方式中，每个象素对应着4位（半个字节），如图2—2所示。在EGA上的CGA部分（即模式0到模式6）这四位是直接映象到RAM中，并且能用直接内存访问（DMA）的