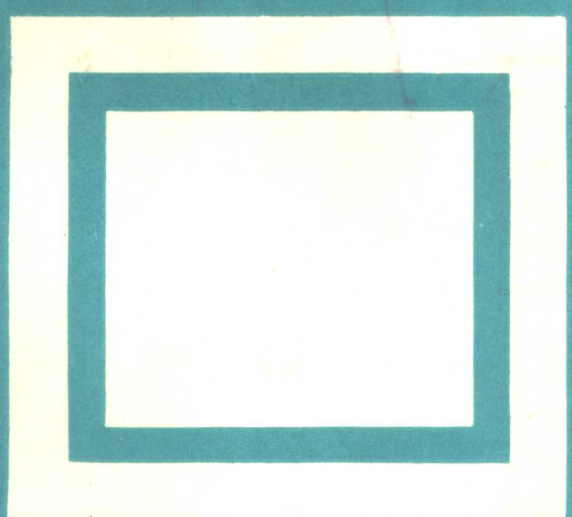
用 Microsoft C 编制图形程序

森林 编



航空工业出版社

用 Microsoft C 编制图形程序

森 林 编

航空工业出版社

(京)新登字 161 号

内 容 提 要

本书介绍了 Microsoft C 语言的编程环境、基本图形元素的生成方法、图形变换方法、以及图表程序的编制方法。本书实例丰富，是迅速掌握 C 语言图形技术的良好工具书。

图书在版编目(CIP)数据

用 Microsoft C 编制图形程序/森林编. —北京：航空工业出版社，1994.9

ISBN 7-80046-816-X

Ⅰ.用… Ⅱ.森… Ⅲ.C 语言-计算机辅助设计-计算机图形学 Ⅳ.TP391.72

中国版本图书馆 CIP 数据核字(94)第 12217 号

航空工业出版社出版发行

(北京市安定门外小关东里 14 号 100029)

北京地质印刷厂印刷 全国各地新华书店经售

1994 年 10 月第 1 版 1994 年 10 月第 1 次印刷

开本：787×1092 1/16 印张：19 字数：487 千字

印数：1—2000 定价：29.00 元

目　　录

引　言

随着计算机软件技术突飞猛进的发展，C 语言编程工具所提供的功能不段增加；但是，不编程工具的功能多么强大，都不能满足所有用户的特定需求。一个程序员很可能要在不同的应用中重复编制同样的代码。为了提高工作效率，编程人员最好建立自己的函数库．本书建立了一些图形显示和数字表示库函数。读者可以用它建立自己的库，并用来设计简单的图形用户界面。

图形库简介

利用程序库编制应用程序有不少便利之处，例如：具有一定的结构和层次的库程序系统可以提供一个抽象的层，使应用程序不受外部环境的影响；提供用户与应用程序之间的通用交互工具；另外，局部变量的使用有利于建立模块化的应用程序。如果在用户界面的所有环节都提供模块化的、独立的库函数，就象大多数 C 库那样，那么就很容易实现高层代码的模块化。移植代码时只要修改库，而不用重写整个应用程序。

正是因为这些优点，人们开发了许多图形库并定义了一些标准，如 CORE、GKS 和 PHIGS，这些标准和相应的库函数过于庞大，不易撑握，不在本书的讨论范围之内。本书旨在教会用户建立自己的简单易行的图形库，以便建立模化的应用程序，并优化程序性能.

开发图形程序的问题之一是如何合理地利用开发平台的资源，如内存和磁盘空间。虽然，微机硬件技术水平发展迅速，但是微机图形程序编制人员总觉得系统资源不够用。这是因为用户对应用程序的要求也在不断提高，从而使得图形应用程序对硬件资源的需求不断增长。因此，如何充分合理地利用硬件资源是一个永远是一个值得探讨的问题。

本书几乎包括所有的非实时图形应用，象字处理，工程设计工具，说明性的和表示性的图形。这种图形使用了现在通用的概念和工具：正文、图形的图表。利用它们可以替代那些乏味的直接输出数的程序。Microsoft 图形库适应了这种需要。

Microsoft C 图形库简介

这本书说明如何使用 Microsoft C 编译器（版本 6.0）的图形库。Microsoft C 图形库程序包括三种基本的实用类型（正文、基本图形、图表）和其它各种各样的类型以及控制功能（窗口、颜色和硬件设置等）。这本书的结构的组织反映了图形库的结构，首先介绍一些设置和控制功能，然后详细讨论三类图形。

图形库提供基本的图形功能，经由一般的调用接口实现，不同类型的显示器有不同的分辨率和颜色能力，图形库使用户从和这些显示器进行交互的繁琐的工作中解脱出来，这样，库使用户用尽可能模块化的方式和基于 PC 的图形硬件交互，不受不同的硬件开发平台的影响。

库提供的技巧能够把输出区域限制成屏幕上一个特定的长方形，并且自动把输出送到这个区域。输出执行的方式独立于不同硬件的性能。坐标系和颜色系统可用现实生活中的方式表示，由库转换为硬件能识别的方式。高层控制可以产生图形显示，包括条形，线形和扇形图表。

其它

这本书除了介绍 Microsoft C 图形库以外，还让读者学会如何使用一个库。读者可以根据库中的程序和把模块组合高层代码的技巧建立自己的实用程序库。作为例子，本书根据 Microsoft C 图形库建立了一个用户界面库。

硬件和软件配置要求

为了使用 MicrosoftC 图形库，用户需要配置一台 IBM PC 或同系列的机器（XT，AT 等），运行 MS-DOS 操作系统（版本 3.1 或更高）。还需要一个图形显示器，尽管图形库可以支持不同类型的显示器，本书给出的大部分例子只适用于 VGA 显示器，一般的概念适用于各种类型显示器，但有一些设置细节不一样。

读者须知

本书假定读者会用 C 语言编程。除此之外，在阅读本书之前，读者必须熟悉 MS-DOS 的 Microsoft C 编译器提供的程序环境。Microsoft C 编译器（从版本 6.0 开始）提供了一个集成式开发环境（IDE），称为程序员工作台，这种基于菜单选择和按键命令，面向项目的环境类似于其它流行的开发系统的 IDE（如 Borland 的 Turbo C 和 Think TechnologiesLightspeed C）。程序员工作台还提供传统的面向工具的开发方式，有一个经过修改的 make 工具（现在叫 nmake），它现在更接近于 UNIX 形式的 make 工具。

本书假设读者习惯 UNIX 系统的工作方式。书中给出了大量源代码例子和相应的 make 文件。

本书作用

本书提供了运用 Microsoft C6.0 及其图形库编制图形程序的入门知识。读者将了解到有关图形开发的开发工具图形库的本质和如何利用图形库开发图形应用程序。另外，读者还会得到 MicrosoftC 图形库开发的用户接口库的完整源代码。

本书对很大范围的读者都很有帮助，主要包括：

学习图形编程的 C 程序员。如果读者是对非实时图形感兴趣的 C 程序员，例如字处理，工程设计工具，说明性和表示性的图形，这本书很适合。在本书里，读者会找到创建正文、图形和图表的要点。

即使是一位熟练的图形程序员，学习复杂的 Microsoft C 图形库也是件并不轻松的事情。这本书将示范如何使用图形库，包括基本图形、图表、窗口、颜色、硬件设置及其它更多的内容。

需进一步了解图形工具的读者，如果已经掌握了图形库的基本知识，读者一般都想自己创建库。本书的最后一部分给出了 SUI 用户界面库，包括完整的源代码和使用它的例子程序。

第一章　库和工具概述

图形功能逐步完善的计算机越来越重要。同时计算机用户的期望也在普遍增加：实现可感觉到的图像的程序更有吸引力，比没有图形的程序更给人以深刻的印象。不管用户是在开发商品还是为自己编程，把自己的想法在程序中用直观的图形表示出来总是很有益处的。

Microsoft C 图形库提供了在程序中实现图形的基本工具。不是每一个程序都需要图形功能，但许多程序利用图形库增加了图形功能后更受欢迎。如果开发象图形界面这样的程序，用户就需要更强功能的图形库，或创建自己的特殊库程序。

利用 Microsoft C 图形库，用户可以给自己程序加上方便的图形功能。图形库基于高级图形程序使用的基本概念。本书将简单介绍这些概念。

1.1　图形库适用对象

如果正在阅读本书的读者是买了 Microsoft C 丛书并经常使用的程序员，图形库就显得非常适合。库提供基本图形功能，和一个独立于硬件特性的接口层，它使得在程序中实现基本图形功能更为容易。

图形库适用于作为独立的 MS-DOS 应用程序的 C 程序。它不能和 Microsoft Windows 或 DS／2 一起使用。这些环境提供它们自己的图形功能。事实上，图形库适用于运行在裸 MS-DOS 系统上的应用程序，而没有任何多余的环境。

1.2　图形库能做什么

除了一般的图形功能，图形库实现了一组表示图形程序。"表示图形"是指把数据用图表示的输出。例如，列在表里一组数据令人厌烦，需要不少精力进行仔细考虑，当同样的数据用扇区图表，条形图表或其它形式的图表表示时，直接可以看出直观的意思。用可见方式表示的数据很容易被理解和使用，且常常比简单的一列数据指示出更多的比较明显的信息。

图形库为实用程序员提供一般图形和表示图形功能。低层功能使用户能画点、线、多边形及各种形式和模式的曲线。表示图形让用户把数据转换成各种类型的图表。

1.3　图形库不能做什么

图形库可以看作是一个完成图形基本功能的应用程序系统；把用户应用程序和硬件平台的实际特性分隔开来的隔离层；使用户的基本图形程序可以在不同 MS-DOS 环境间移植的方法；把冗长的数据转换成完美直观的图表表示的工具。

用户无法利用图形库编制复杂的图形程序，象三维物体的显示、动画、加亮、旋转等

等。图形库并不准备为各种用户提供所有图形功能，也不替代复杂的图形程序包。

1.4 Microsoft C 环境

这部分包括有关在 Microsoft C 环境下编程的一些知识，以及如何利用 Microsoft C 的工具在应用程序中调用图形函数。本书不对 Microsoft C 和它的工具作详细介绍，读者应该通过其它途径熟悉这些知识。当然，如果读者在这方面完全不熟悉并正要开始学习，这部分会有一定帮助，这里还包含了本书中讨论的图形库和例子的背景。

如果读者对 Microsoft C 很熟悉，为了使用图形库程序，应该把库程序包含在库 GRAPHICS.LIB 中。如果用户在安装编译器时没有把图形库程序包含在标准库中，那么就应该显式地告诉链接器把 GRAPHICS.LIB 作为用户 build 程序要寻找的一个库。

1.5 集成式和面向工具开发环境

C 语言来源于 UNIX，而 UNIX 系统下的程序员习惯以面向工具的方式工作，即利用一些独立的工具，单独使用或者和其它工具组合使用，进行工作。这种面向工具的方式也被引入到其它平台中。一些制造商却使集成式开发环境广为流行，象 Borland 为 IBM PC 系列开发的 Turbo C，ThinkTechnologyies 为 Apple Macintosh 开发的 THINK C。使用这样的系统，当前项目总是处于活跃状态，对其操作可以通过菜单选择和接键命令进行。集成式环境提供直接对项目进行的操作（如编译，链接，调试等）。

Microsoft C 传统上采用面向工具的方式。（该公司的 QuickC 提供集成式开发环境。）在公开发行的 6.0 版本的编译器中，Microsoft C 同时也包含了集成式开发环境，叫程序员工作台。

哪种方式更好，面向工具还是集成式？这依赖于许多因素，包括私人观点。就个人而言，我习惯于面向工具方式。它更适合我，使我能够在一些不同环境中使用同类方式。同时，使用面向工具方式可以更容易地设计程序例子，本书例子就采用这种方式实现。

Microsoft C 有四个基本工具。第一个是在 MicrosoftC 外部使用（除非用户使用集成式程序员工作台环境）的有关文件的创建和操作的一组工具；第二个是编译器，用来把 C 程序转换成目标文件。第三个工具由链接器和库工具组成，对目标文件进行操纵。第四个是 make 工具，实现程序的自动编译和链接。下面对这些工具作进一步介绍。

1.6 外部工具

第一个工具是在 Microsoft C 外部对 MS-DOS 环境进行操纵的一组功能。包括管理磁盘目录结构的基本 MS-DOS 命令：拷贝，删除，重新命名文件，编辑源文件。

1.7 编译器

编译器把源文件转换成目标文件，目标文件用来（通常和其它目标文件结合在一起）创建实用软件．编辑器实际上是几块程序共同工作，包括宏预处理器，分段编译器，装配器．每一块都可以单独使用，但更好的方法是，用编译器作为驱动程序协调控制这几部分的工作，把 C 源代码转换成目标代码．

该驱动程序叫 CL，启动时提供源文件名及一些选择项．选择项前面有一斜杠（/），就象 MS-DOS 的一般格式．也可以在选择项前面加一水平杠[-]，使其它 C 环境下的程序员容易熟悉．有许多选择项可供使用，表 1-1 列出了一些最重要的选择项．

表 1-1 常用编译选择项

/C 只编译，不链接	如果用户不加这个选择项，CL就试图根据源文件生成可执行文件
/O 定义预处理符号	/OSYM定义一个空值符号常量；/OSTM＝Val定义一个值为val的符号常量
/AM 选择存储模式	m可为T，S，M，L，H，分别表示微型、小型、中型、大型、巨型模式
/Z	生成供Codeview调试器使用的调试信息文件，链接时必须有/CO标志

读者应该自己学习使用 CL 命令的选择项，包括选择项的功能和使用时的语法．掌握了基本格式之后，方能看懂本书后面的例子．

1.8 目标文件工具

第三个工具是操纵目标文件的一组命令，操作目标文件的目的是把它们转换成可执行的应用软件．两个工具用来实现这个目的：目标库管理器和链接器．

"目标库管理器"对把许多目标文件作为一个整体操纵很有用，本书后面将举例用系统的库程序实现一个用户界面库，其中要用到很多库程序，最好把它们作为一个整体看待．

"目标库"就是把许多目标文件组织成一个文件．目标库类似于小型文件系统：里面有目标文件目录，也有目标文件本身．

链接器可以从目标库中取出链接好的目标文件需要的部分．这样，当用户链接一个库时，并不用链接上库中所有程序，事实上许多程序并不需要．

1.9 LINK

"链接器"把目标文件组合起来生成可执行文件（用户的实用软件），并可以提供地址映射和调试信息文件供调试工具使用，如果需要的话．运行链接器有多种方式，最简单的是直接敲入命令：

```
LINK
```

并回答提出的问题。当需要指定目标文件时，用户输入的文件名可以有.OBJ 扩展名，也可以没有。每个文件名用加号（+）隔开，如果当前行空间不够，以加号结束一行就可以继续回答同样的问题。

下一个问题询问要产生的可执行文件名。缺省回答在方括号内显示，直接接回车键就表示采用这个名字。随后要回答存放链接器输出（调试消息）的文件和需要与程序链接的库的名字。每个问题用户都可回答多个文件名。最后一个问题询问"定义文件"的文件名，用户可以把特殊的链接消息存放在这个文件里。

在交互过程的任何时候，用户敲入分号就结束这个过程。(链接器用缺省回答作为没有回答的问题的答案。）为了忽略输入或者输出，用户可以用 NUL 作为文件名。例如，用户若要链接文件 x.obj并生成可执行文件 x.exe，与 LINK 的交互过程如下:

```
C> LINK
Object Modules [.OBJ]: x.obj
Run file [x.exe]: x
List file [NUL.Map]: NUL
Libraries [.LIB]: graphics
Definitions File [NUL.DEF]:
```

用户也可以把回答列在 LINK 命令行上，按照它们被询问的顺序，如下例:

```
C> LINK x,,, graphics;
```

注意被忽略的参数采用缺省值，用分号结束命令行。一个实用软件经常包括许多源文件，MS-DOS 的命令行上根本容纳不下。解决这个问题有两个途径。第一，用户可以把一些目标文件组织到一个库中，库中只有少数几个目标文件（如主程序）可以被链接。或者，用户可以使用回答文件。"回答文件"包含了与链接器交互时的所有问题的答案。为了让链接器知道从回答文件中取答案，回答文件名前加@标志，下例告诉链接器从回答文件 x.link 中取答案:

```
LINK @x.lnk
```

1.10 Make 工具

Make 工具根据成份规格说明生成可执行文件。成份规格说明用存放在一规则文件中的一组依赖关系说明和规则描述，通常称为 MAKEFILE。(注意：Make 工具，象 Microsoft C 一样在 UNIX 环境下有它自己的根（root)）。包含在 Microsoft C 早期版本中的 Make 工具只是在总体上与 UNIX 下的 Make 程序基本相容，这使使用多重环境的用户感到很不方便，而 Microsoft C 包含一个更类似于 UNIX 下的 make 程序的 make 工具，称为 nmake。这样，极大地方便了用户。本书中的例子使用 nmake 文件，如果用户习惯于使用其它形式的 make 工具，则需对其稍作修改，但仍能说明例子需说明的内容。

尽管 make 文件可能有多种不同的结构，基本上是由三个重要部分组成：赋值、依赖关系说明和规则。"赋值"给变量定义一个替代串，例如，下例定义了一个变量

"CFLAGS"，其值为-DOS__MS-DOS

```
CFLAGS=-DOS__MS-DOS
```

用圆括号把变量括起来并在前面加上"$"符就得到该变量的值。因此，如果预先给出了上面的赋值，下面的第一行就会被自动解释成第二行:

```
cl /C $ (CFLAGS) source.c
cl /C -DOS__MS-DOS source.c
```

"依赖关系说明"用来描述源文件和生成文件间的时间次序关系，"生成文件"就是通过某种途径生成的文件（例如一个目录文件可通过编译一个源文件产生）。依赖关系说明几乎总是意味着生成文件由源文件生成。例如，用户有一个包含头文件 x.h 的源文件 x.c，只要用户修改了 x.c 和 x.h 中的一个，就要重新进行编译。这可以在 make 文件中加入下面的依赖关系说明描述:

```
x.obj: x.c x.h
```

这行代码表示 x.obj 依赖于 x.c 和 x.h。在 make 工具处理到这行依赖关系说明时，它要保证 x.obj 比两个源文件更新。在冒号前可以有多个生成文件。冒号表示所有生成文件依赖于随后列出的源文件，任何一个源文件在 make 文件中也可作为另一依赖关系说明中的生成文件。make 工具依次处理这些说明。在运行 make 时，用户给出需生成的生成文件。如果缺省，就生成 make 文件中第一个依赖关系说明中的生成文件。

"规则"是触发依赖关系的命令。规则紧跟在依赖关系说明行后并以一个制表符开头。可以有多行规则，如果下一行是空白行或是下一个依赖关系说明，则表示规则结束。在前面的例子中，x.obj 的规则可以是:

```
x.obj: x.c x.h
cl /C $ (CFLAGS) x.c
```

make 工具也包含生成某种特殊格式的文件的功能，这要使用 make 文件扩展功能。用户可以在 make 文件中加入新的规则并定义许多自己的参数替代串。本书中的例子不使用 make 的扩展功能，因此这里不作进一步讨论。用户如果发现有不理解的地方，可以查阅 Microsoft C 手册。

1.11 包含图形库

Microsoft C 的 INCLUDE 目录下的 GRAPH.H 包含了一组定义（预处理常量，有用助记名和函数格式）。这个 INCLUDE 文件应该被所有使用图形函数的程序包含，其中的定义在本书中将频繁地提到。GRAPHICS.LIB 包含了图形库的所有库程序。同样，有关表示图形的定义包含在 PGCHART.H 中，库程序在 PGCHART.LIB 中。

在安装 Microsoft C 时，用户可以选择是否把目标库连接到编译程序的正常查找路径上。如果已连接，所有图形库程序可被自动链入。如果选择不链接，用户必须显式地包含

库。在这种情况下，用户要编制一个使用了图形库程序的软件时，为了找到库程序，必须告诉链接器查找图形库。

可以用下面的链接命令生成一软件:

```
LINK x,,, graphics
```

如果用户链接时使用回答文件，只要把库包含在问答文件中即可。链接程序 x 的回答文件如下:

```
x
x
x
GRAPHICS;
```

1.12 库程序使用

本书讨论了每个库程序，给出调用方式和返回信息。一些不太重要的细节常常被忽略，举例来说，本书没有指出处理器的模式，比如地址用 far 指针还是 near 指针，或者是 far 或 near 指向 far 或 near 的指针，大部分情况下，这些信息只有在确定参数是否匹配才有用，而用户根据库中头文件说明的函数格式调用时根本不会出现参数不匹配问题。(用户想知道这方面的详细信息，请查阅 Microsoft C 手册。)

有些库程序直接返回状态信息，有些则不直接返回。通常，用户可以调用_grstatus 得到最近调用的图形函数的状态信息。该函数没有参数，返回一个整数状态值（状态值在 GRAPH.H 中定义)。0 表示调用成功，负数表示调用错误，正数代表警告信息。

1.13 工具总结并举例

前面只是一个如何使用工具的简单介绍。许多情况下，命令有许多选择项影响工具的作用。如果用户是初学者，在阅读完前面的介绍以后，还要查阅介绍这些工具的有关资料。

现在，我们给出一个介绍工具和如何使用图形库的例子。同时给出了本书中例子的形式。通常，例子包含源代码，有时也列出相应的 make 文件（用来生成执行文件)。如果 make 文件和前一个例子中的一样（除了几个文件名的变化)，就不再列出，但是，用户如果有本书的源文件盘，每个源程序都有相应的 make 文件。

这是一个传统的 C 程序例子，显示"Hello, World"。程序 1-1 在屏幕中间显示该信息，等待一个按键动作，然后退出。程序 1-2 是该程序的 make 文件。

程序 1-1 打印 Hello, World

```
/ * hello.c - Print "Hello, World" on the screen using graphics functions.

Note: assumes font files are in C:\FONTS\
```

```
 * /

#include <stdio.h>
#include <graph.h>

#ifndef   TRUE
#define   TRUE 1
#endif
main( int argc, char * *argv ) {

    if ( ! _setvideomode( _MAXRESMODE ) )
        fprintf( stderr, "Can not initialize to _MAXRESMODE.\n" );
    else {
        / * Setup a convenient coordinate system * /
       _setwindow( TRUE, 0.0, 0.0, 1.0, 1.0 );
       / * Go to somewhere near the screen center. * /
       _moveto_w( 0.3, 0.5 );

       / * Set a visible color. * /
       _remappalette( 1, _BLUE );
       _setcolor( 1 );

       / * Register font * /
       _registerfonts( "C:\\FONTS\\HELVB.FON" );
       / * Setup desired font * /
       if ( ! _setfont( "t'helv'" ) )
        _outtext( "Can't init font. \n" );
       else
        / * Output string of graphics text. * /
        _outgtext( "Hello, World!" );

        getchar();

        / * Reset to original graphics mode * /
       _setvideomode( _DEFAULTMODE );
    }
    exit( 0 );
}
```

程序 1-2　程序 1-1 的 MAKEFILE

```
# MAKEFILE for hello world example program (hello.c)
#

LDFLAGS= /NOE
CFLAGS= /AL

all: hello.exe

hello.exe: hello.obj
           link $(LDFLAGS) hello...graphics;
hello.obj: hello.c


hello. obj: hello.c
```