微机接口与应用系列丛书　　学苑出版社

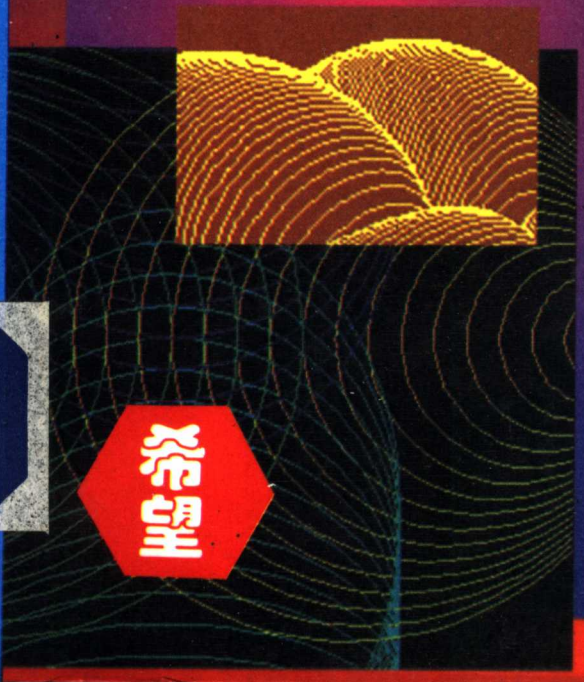
John L. Bradberry

SAM
PUBLISHIN

COMPUTER GRAPHICS ENVIRONMENTS

计算机图形环境

Algorithms for Design and Control



- Learn graphics device interfacing for hardware and software
- Detailed algorithms and examples in pseudo-code
- Develop your own eye-catching displays and screens

Disk contains complete solutions in the C programming language

微机接口与应用系列丛书

Computer Graphics Environments

计算机图形环境

John L. Bradberry 著
宋云生 彭振云 徐广方 译
吴兴龙 张景生 审校

学苑出版社

(京)新登字 151 号

内 容 简 介

本书通过对一个通用图形库 DIGL 的设计和实现进行详细描述，全面介绍了图形学的基本概念、图形设备接口技术、图形软件的层次结构和窗口程序设计方法。全书由九章和八个附录组成。

本书可供从事图形设计与应用的工程技术人员使用，也可作为大专院校相关专业的参考书。

欲购本书的用户，请直接与北京海淀 8721 信箱书刊部联系，邮政编码：100080，电话：2562329。

版 权 声 明

微机接口与应用系列丛书

计算机图形环境

著　　者：John L. Bradberry
译　　者：宋云生　彭振云　徐广方
审　　校：吴兴龙　张景生
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：兰空印刷厂
开　　本：787×1092　1/16
印　　张：13.75　　**字数**：307 千字
印　　数：1～5000 册
版　　次：1994 年 5 月北京第 1 版第 1 次
ISBN7－5077－0803－9/TP・14
本册定价：29.00 元

学苑版图书印、装错误可随时退换

前　言

近年来，随着计算机体系结构和语言的发展，用户和应用程序之间的通讯方式也发生了戏剧性的变化。传统的用户界面基于简单的文字对话，现在已被复杂的三维按钮选择所取代，这归功于窗口环境。同时，超高分辨率的多彩色工作站显示器也取代了低分辨率的单色终端。计算机的处理能力和速度在提高，而价格相对稳定，自然用户的期望也越来越高。今天的用户希望数据的可视化更具有真实感，更能反映细节。生成、控制和表现高分辨图形的能力已成了每个计算机程序员的开发包中必不可少的特性。

过去，编写用户界面和计算机图形算法被认为是和编写应用程序不同的技艺。图形学领域一直很狭窄，而且高度专门化。现在，随着桌面印刷和 CAD 的发展，图形学已经走向实用。用户对传统分析应用程序的要求仍在不断提高，却不应再把图形和用户界面看作一个孤立的部分。

在今天的市场上，通过模拟和生成图形图像使数据达到可视的能力已变得必不可少。同样，生成和控制信息表像的能力也变得必不可少。

这本《计算机图形环境》包括：

- 从硬件和软件角度介绍图形设备接口技术
- 多层次图形接口——从矢量原语到复杂的高层多边形变换
- 菜单接口，包括文字和图形菜单接口设计
- 用伪代码写的能用于任何计算机语言的算法和用 C 语言写的完整的应用软件实例，并提供一张软盘。

图形标准

有多种图形库和图形标准供选择。编译器厂商和第三厂商的图形库也为应用开发者提供了很多精妙的图形运算。但是，任何图形库都无法包罗万象，因为有三个主要问题：

- 你总是被限于一个特定版本的编译器、操作系统、一组有限的图形操作，或上述三种情况的某种组合。
- 图形库并不支持你最感兴趣或将来可能要用的图形设备。
- 图形库只能由提供者扩充，而且一般没有源代码供你修改。

GKS(Graphics Kernel System)和 PHIGS(Programmer's Hierarchial Interactive Graphics System)这样的图形标准试图通过限定语法和图形原语来解决这些问题。的确有用这些标准实现的图形库，但也有两个问题：

- 标准(按定义)有助于强调应该做什么，却很少或不谈怎样做。在实现标准时，必须保证与语言和系统无关，因此，标准的实现是不同的。
- 为了满足不同用户的需求，标准往往只取最少的公用特性。比如，为了向前兼容往往会影响标准实现的性能。

本书提出的 DIGL 图形库试图通过以下三种机理避免上述问题：

·所有源代码和应用程序代码都随着 DIGL 提供，并解释做什么，怎样做和为什么这样做，还提供了设计和调试信息，甚至提供了一个关于错误诊断的附录。

·为了能扩充图形设备的支持和能力，DIGL 要求编写附加设备驱动程序。以库的形式组成 DIGL，提供给用户几个库和支持工具，帮助用户定义自己的图形运算。

·因为不能要求读者去搞清 DIGL 的所有细节，所以主要通过例子来解释。书中提供了完整的实例，通过这些程序，读者至少能看到自己所需要做的大部分工作都已提供了源代码。

阅读本书的先验知识

本书不需要图形学的先验知识，但必须熟悉程序一级的计算机实现。书中有些关于矩阵运算和多边形变换的理论讨论，因此，如果你想修改关于这些运算的算法或源程序，则必须具备基础代数和分析几何方面的知识。不过，大多数情况下，算法都能照原样使用。

建议有一些 C 语言编程知识，但并不一定要有。所提供代码可用你钟爱的任一种编译器编译。本书程序实例适应两种最流行的计算机体系：MS－DOS 和 UNIX。附录 C 和 D 提供了关于这种体系的提示和建议。附录 G 详细提供了关于在其他计算机体系下运行的图形信息。

对于大型 C 应用程序，你可能需要修改一些低层驱动程序以适应编译器的差别。本书在给出代码的同时，还对程序的修改时间和方式作了解释。

本书约定

本书采用如下印制约定：

·程序代码，命令，语句，变量及任何从屏幕上看到的文字都用计算机字型。

DIGL 只展示了当今图形学应用的一小部分，但是，对这个优美的系统，你可以任意修改。我们希望你能让它得到更完美的展示。

本书主要讲述了图形和菜单生成技术，通过研究这些源代码实例，你显然能设计出可满足自己特定要求的系统。

译者的话

《计算机图形环境》一书是美国佐治亚州技术研究所理科硕士 John L. Bradberry 的最新佳作。它从硬件和软件的角度，系统地介绍了图形设备接口技术、多层次图形学接口、菜单接口以及用伪代码写的能用于任何计算机语言的应用软件实例。本书内容充实，数据可靠，概念明确，图文并茂，深入浅出，通俗易懂。书中提供的大量程序和图表实用性强，有的可以供用户直接使用，有的只要稍加修改就能满足有关用户的实际要求。本书可以作为从事计算机图形学研究和开发的工程技术人员的工具书，也可以作为大专院校有关专业的教科书和参考书。

参加本书翻译的有彭振云(1—4 章)、宋云生(5—9 章)、徐广方(附录 A—H)，吴兴龙同志审校了全部译文，张景生同志对本书翻译给予了热情的帮助和指导。

由于我们经验不足，水平有限，时间仓促，因而译文的错误和不妥之处在所难免，恳切希望广大读者批评指正。

译者

一九九四年五月于北京

目　录

第一章　系统结构、语言及应用

几乎从计算机诞生之日起，就出现了对控制单个像素(Pixels，即图像元素或字符点阵中的点)的希望和要求。需要图像和图形显示的应用在研究和开发实验室开始占据主导地位。在过去二十年中，图形学领域采用了多种系统结构，所取得的效果各不相同。这些结构对软件要求和应用性能都有影响。

1.1　图形系统结构和技术

各种不同的应用有不同的图形系统结构，各有优缺点。本章介绍两种最常见的：分布式图形系统结构(主CPU加外设节点)和集中式图形加速器系统结构(有时称为专用图形加速器)。

1.1.1　分布式图形系统结构

图1.1是一种最普通的图形系统结构。

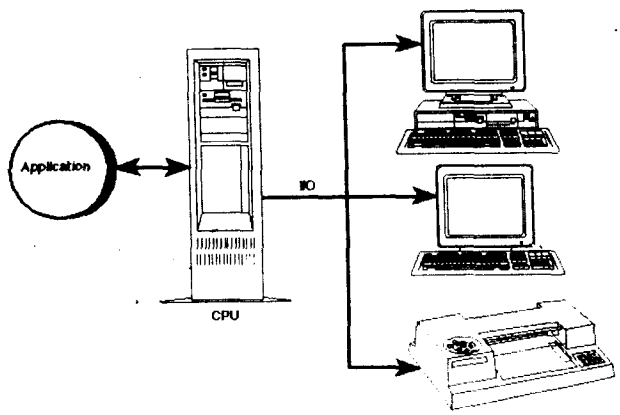


图1.1　分布式图形系统结构

在这种系统中，一个或多个图形应用间接地接在一个远程图形外设上，虽然可以将成百上千的不同图形终端或绘图设备接在小型计算机或工作站上，但通常同时只接几个。不过，这种结构对于图形应用编程来说设计上有些困难。

• 图形协议

图形协议(Graphics Protocol)指完成图形操作(如从A点到B点画一条线)所需的字符序列。可以猜想这些操作对不同图形设备各不相同。终端典型地使用非打印ASCII字符，如<ESC X,Y>，而HP数字绘图仪这样的设备则采用简单的ASCII命令，如PA1234 ,45 67J。采用何种设备协议通常由图形库确定。图形库可能是图形应用程序的一部分，或者由第三厂商

(CPU 制造商)提供。

要注意的是,这种接口往往对语言和操作系统独立。任何能完成非格式化 I/O 操作的语言都可用于控制这种外设。命令解释过程将在本书后面详细介绍。

· I/O 接口

I/O 接口机制在图形设备,计算机和应用程序之间起物理和逻辑连接作用。物理通道可能是串行连接(如终端),IEEE 连接(如,绘图仪),或与 CPU 板的专用高速直接连接。

图形库处理图形协议中的差异,还处理与设备的逻辑连接,如计算机地址映射或运行时间。

在串行通信中,可选参数是非常简单的,比如,波特率(每秒串行传输的比特数,记为 bps)和"握手"(Xon/Xoff)。图形库不涉及这种接口的低层,但必须保证字串传输之前设备连接无误。

在更复杂的连接中,如 IEEE 488 GPIB 总线接口或直接总线接口,驱动接口要困难一些。

这种 I/O 接口用到的典型参数是带宽,即产生图形图像的速度。对于简单的矢量图形运算,单个运算也往往需要多达 20 个字符。这看起来不多,但请想一下,在一些小的 3—D 绘图应用中,往往会有多达 50,000 或更多的点! 若采用 9600 的波特率,串行口每秒只能传送 1200 个字符。这样,采用串行连接将要花 13.89 分钟! 由于复杂的绘图所需点数随数量级增多,因此,不难理解为什么工作站图形能持续得到应用。

· 初始化和反馈要求

无疑,不同的图形设备需要不同的初始化过程。有两种不同的初始化——自动初始化和手工初始化。视频设备(如终端)可以认为是自动的:用一个或多个命令将屏幕清除,并将设备置成图形方式。一旦执行完这些命令,就可以认为已置为图形方式,并可以接着显示图像。

对于绘图仪,初始化过程需要在图形方式之前由人或机械手手工装入一迭纸。HP7550A 这样的绘图仪识别取纸命令,但像选笔这样的命令却必须手工完成。这些操作可能还需要反馈,即与应用程序或驱动程序进行双向通信。还有其他一些手工初始化。比如,很多种视频显示器要求用户装入一个新驱动程序,该程序在多同步监视器上选择显示分辨率和同步方式。

1.1.2 集中式图形加速器系统结构

图 1.2 是另一类常见的图形系统结构。在此情况下,没有外部图形设备。图形可通过存储映射 I/O 与运算加速器相连。这种结构比分布式结构快得多,但也有些缺点。

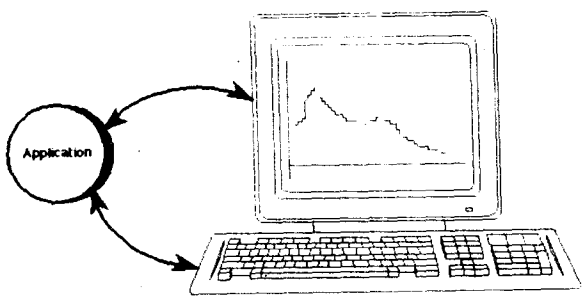


图 1.2 集中式图形加速器系统结构

·图形协议

这类结构和一些重要的图形协议有关。在此情况下，通常要用到一层复杂的图形驱动软件作为接口。和采用外部图形设备不同，图形应用和图形设备之间的连接完全是逻辑的，或通过软件实现。

应用程序不是通过传送代表图形命令或数据的字符串，而且直接和驱动程序进行通信。这样做最重要的结果是，这种协议是和语言有关的。与分布式系统结构不同，所提供的驱动程序基于一种或最多两种语言——C，Fortran 或甚至汇编。除非你所选的语言包含在驱动软件包中，否则必须用混合语言和所支持的语言沟通。

·I/O 接口

由于这种接口高度专门化，所以 I/O 机制是一组不能统一控制的专门操作序列。对于可直接读写的存储映射 I/O。可以专门设计一些图形运算。

带宽由内存读写速度和 CPU 速度共同决定，这意味着这种模型比分布式结构和模型能支持多得多的矢量和光栅操作。这种结构每秒能处理上百万个字符，而不像分布式结构那样只有几千个。这样，在分布式结构中需要十分钟的绘图操作在这里只需要一秒钟就行了。

·初始化和反馈要求

和外部图形设备一样，这种结构对初始化也有特殊要求。自动初始化也要采用。而且也要用驱动程序来处理初始化过程中所有具有特殊要求的操作。

1.2　图形和语言标准

许多图形语言和协议标准已经被采用了多年。早在 70 年代早期，人们就开始着手制定和采用用于下列目的图形标准：

- 减少重复劳动和图形程序的复杂性
- 降低实现图形应用的代价
- 提高图形应用的可移植性

《计算机图形学——原理和实践》(Computer Graphics——Principles and Practice，Fdey et al. 1990)一书从历史的角度对图形标准进行了详细的讨论。以下是这些内容的概况。

ACM 的 SIGGRAPH 小组于 1977 年建立了“3D 核心图形系统”(“3D Core Graphics System”)，并于 1979 年对此规范作了改进，该系统被 ANSI 接受，并得到广泛使用。但是，1985 年宣布的 GKS(Graphics Kernel System)才是第一个正式国际图形标准。1988 年，GKS 的改进型和另一个标准 PHIGS(Programmer's Hierarchial Interactive Graphics System)一起成为进行图形操作作了种种限制。对于 PHIGS，这常常需要特殊的硬件支持。

不幸的是，这些标准的特性并不支持所有的图形操作。而且，并非所有图形设备都能以同样的方式实现同一操作。要制定一个功能强大到能支持很大范围图形操作的标准必须考虑很多因素。

比如，对于笔式绘图仪来说，清屏意味着什么？如果你的设备不支持标准中的某种图形操作怎么办？如何对标准扩充以使其跟上图形学的发展？在采用标准时是否要兼顾性能局限？可见，制定和使用一个能支持大量图形操作的图形标准决非易事。

1.3 图形应用模型

很多程序员都喜欢编写针对一种特定系统结构或图形库的程序，而根本不考虑对其他图形设备的支持。本书打算通过一些例子和方法介绍一种不同的作法。以一种灵活的模型为基础，设计一个分层图形库，然后即可在不同编译器和系统结构下编写出通用、可移植与设备无关的图形应用程序。

1.3.1 DIGL 图形模型

DIGL(与设备无关图形库，Device－Independent Graphics Library)的结构支持很多种系统结构。图 1.3 所示的 DIGL 框图有点像我们前面讨论的分布式图形系统结构。但是 DIGL 的实现假定对于每种图形设备都有一个内部驱动程序。图形设备的接口位于 DIGL 的最底层。其他图形设备可通过加入相应的驱动程序来支持。

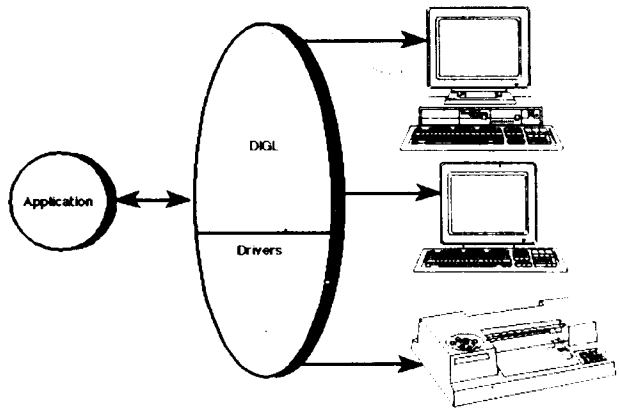


图 1.3 DIGL 系统结构模型

该库的结构基于本书中讨论的程序实现。结构化程序设计对于软件扩充和故障排除是必须的。自顶向下软件设计方法已变得有点陈旧。自顶向下编程通常要用自底向上实现来补充。这种软件分层的结果是一个倒金字塔型(图 1.4)。

较低的层次实现一些与设备和 CPU 有关的关键功能。图形设备驱动层(第 2 层)可以用第三厂商的软件。此结构的下列三个特性使用得系统具有可移植性：

· 随着此结构的实现，上两层一直用通用代码，在此情况下是 C，这些层次中所有代码都与应用和 CPU 无关。

· 每层都可以通过简单地加入绘图算法或图形设备进行无限制的扩充，而且对已有代码几乎没什么影响。

· 按照这种倒金字塔结构，那些与设备和 CPU 有关的部分在整个系统中要保证占有最少的代码。对于一个多达 100,000 行的源程序，一般与 CPU 有关的部分应少于 1%(金字塔底部)。

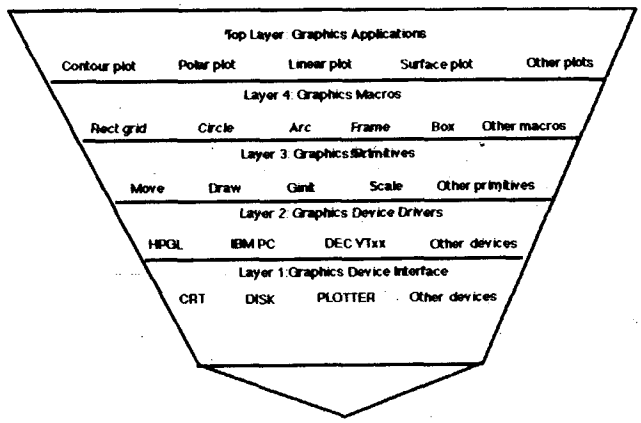


图 1.4　DIGL 模型的结构

1.4　小结

本章介绍了两种基本图形系统结构(分布式和集中式)。在编写应用程序时要考虑的一些设计问题。

即使没有图形标准,也能通过建立基于一个灵活通用模型的分层图形库来编写通用、可移植并与设备无关的图形应用程序。

第三、四、五章介绍特定图形操作的技术问题,着重介绍 DIGL 模型的开发。

第二章 图形学基本概念

第一章"系统结构、语言和应用"介绍了两种图形系统结构。但并没有谈到像 IBM，Apple 和 DEC 这样的专门计算机厂商，因为我们的目的是想给出一个整体概念。

本章将细化 DIGL 模型，并介绍第五章中的图形系统所需要用到的图形运算。

2.1 问题和答案

这一章要介绍关于图形学术语和运算的背景知识，为此，先回答一些读者可能会提出的问题。

1. 我只需编一个简单的绘图程序，还一定要了解这些知识吗？

提这些问题的往往是那些希望不费太大劲就能解决问题的读者。实际上很多情况下图形学要求(和其他软件需求一样)开始的确很简单。遗憾的是这种情况不会持续太久，若干天、若干星期、甚至若干月以后，就会出现新的要求，如"语言系统用于 IBM 机器，如果在 Mac 机器上应用，那么会有多少问题呢？"

我们在此试图给出尽量广泛、充分的细节，以便读者将来能进行修改。如果你觉得你的要求的确很简单，那就是只须用 DIGL，而不必须涉及其实现问题。

2. 哪种计算机平台最适合图形应用？

"计算机平台"(Computer Platform)是程序员用来开发和运行图形应用的硬件和软件工具的总称。至于什么样的平台是最好的，则取决于你的要求、经费、时间和对所选系统掌握的程序。

为了选择最合适的平台，你得了解所用程序语言和拟选硬件系统的局限性，然后开发或选用一个图形库，以使你用尽可能简捷的方法解决问题。

没有一种图形语言或规范是完美的，因为图形应用和可再生的图像一样，其数量和种类是无限的。

任何试图适合一切应用和用户的计算机语言和图形学规范都是不会成功的，要么语法太复杂，要么实现太简单。虽然这听起来有点武断，但过去三十年中计算机的发展证明了这种观点。

3. 设备无关性的真正含义是什么？

设备无关性(Device Independence)指一个图形程序能在多种图形设备上运行，产生同样的结果，而且不需要修改程序。在 DIGL 模型中，DIGL 将程序翻译成一组与其支持的各种图形设备兼容的命令。

4. 可移植性和设备无关性是否会使图形应用速度降低？

这一问题主要是考虑到高层宏指令或更直接的内联指令的使用。通常高度可移植性(Portability)和设备无关性意味着要增加附加指令。

调整 CPU 和高效率硬件的发展趋势鼓励发展更通用的应用软件。按今天的标准，4—

8MHz 的 CPU 是很慢的，现在市场上 33－40MHz，甚至更快的 CPU 都很多。

5. 我所用的第三厂商的图形库具有设备无关性，为何还要开发？

很难说服人们不去开发给定图形库中已提供的程序。在许多情况下这没有必要，针对某种计算机的多种编译器的图形库往往都会说明所支持的设备数量和类型。本书的 DIGL 模型提供了逻辑、框图、算法和代码，利用这些读者可以开发和修改自己的 DIGL 版本。第三厂商的图形库很少会提供源代码和充足的支持，因此开发者很难做到这一点。

6. GUI 只是 Windows 的另一个名字吗？

GUI(图形用户界面，Graphical User Interface)一词可以和 Windows 互换。但 GUI 概念的内涵比这广泛得多。Windows 可借助于扩充图形字符(如 IBM PC 字符集)以文本形式出现。对于位图窗口界面，则必须高速处理单个像素。这两种窗口界面分别在第六章和第七章中讨论。

7. 面向对象的技术能解决所有的问题吗？

面向对象编程(Object－Oriented Programming，简称 OOP)是一种组织和存取数据的技术，它能使应用本书至此讲到的原理更加容易。第八章详细讨论 OOP，要记住在图形运算中采用 OOP 技术能大大提高产品性能。

2.2　计算机平台

建立图形开发环境时需选用计算机平台，每种系统结构都有优缺点，尤其要注意的是操作系统的局限和开发环境支持，因为它们会影响应用软件和图形库的开发。

2.2.1　大型和小型计算机平台

按一般规律，大计算机(从小型机到巨型机)总是将图形和其他驱动任务分配给较小的节点，即集中式 CPU。显示器可以是远程低分辨率终端或采用专用驱动程序的高性能视频显示器。操作系统是专用的，而制造商的开发支持是关键因素。软件库大而复杂，程序的编译、链接和运行需要几个额外的步骤。当然其优点是主 CPU 部件速度较快。

2.2.2　低价/高性能的工作站

工作站可看作是一种中间层次的计算机平台。流行的工作站(如 Sun SPARC 系列)拥有能与低档小型机媲美的处理能力。而从使用角度看又像个人计算机。工作站有为超高分辨率视频显示设计的专门图形硬件。

工作站采用 UNIX 作为主要操作系统，能保证提供各种应用程序和充分的开发环境支持。但有些程序员可能会感到为难，因为工作站不支持 C 和 C＋＋之外的语言，这是 UNIX 的特性所致。像 Fortran77 这样的语言在工作站中都让位于 C。

在图形编程中，必须用到 Windows 界面。但在开发通用图形库时，像 X Lib 这样的窗口环境和由厂商提供的专门环境(如 DEC、HP)可能会带来一些问题。第六、七章和附录 D 讨论了这些问题。

2.2.3　PC 机——最廉价的方案

PC 机这样的微型计算机允许对显示特性进行更直接的控制。和工作站一要，流行的 PC

机也提供大量第三厂商应用程序。这种平台的主要缺点是编译器和程序员过份依赖于系统专门的特性。这样,即使仅仅改用编译器也要付出艰辛的劳动来移植应用程序和软件库。

2.3 编程语言支持

计算机语言为应用程序员提供了很大的参考余地。图形要求可以抽象为一个基本功能子集或低级运算。不幸的是,一般的计算机语言都还不具备图形运算能力。

2.3.1 性能局限和要求

在任何一种现代计算机语言中,图形设备类型通常都不作为标准项。Fortran、C、C++、Ada、Pascal、Prolog、Lisp、Forth、FP 等语言都是为各种目的开发的语言,如科学计算,逻辑编程,人工智能。这些语言大多数都有某种形式的内部 I/O 功能,可用于文件和 TTY 等设备。

在第四和五代语言中,面向对象的技术作为标准组成,其 I/O 设备类型模型必定要扩大,以提供更多的图形支持。不幸的是,支持一组基本运算的通用图形设备并不是 I/O 软件包的一部分。原因之一是因为图形设备种类太多。有趣的是 Windows 标准(如 X Lib 和 Motif)比本书讲到的 DIGL 对图形运算的要求更高。

没有标准就意味着应用程序员必须开发自己的图形设备类型。这通常会使用性能稍差一点的图形,而且图形运算的范围也很有限。

2.3.2 汇编语言或专用硬件?

为了充分利用专用图形系统的性能,还有些不是很吸引人的做法。用汇编语言或专用硬件几乎总能实现比高级语言更卓越的性能。但是,这种程序调试很难,移植性也比较差。这些缺点几乎无法用所实现的高性能来补偿。

2.4 图形原语

现今所用的通用计算机语言都不能用本身的语言功能来实现图形应用的要求。必须在软件中加入通用图形设备类型。因此,最好的办法是构造一组用于开发图形库的低级、通用图形运算。

第一章所述的 DIGL 模型第三级(图 1.4)包括一个称作图形原语(Graphics Primitives)的中间层次。图形原语可看作是一些低级指令(move 和 draw),可用来组成高级宏操作,如 reactangle 和 circle。

表 2.1 列出了一组基本图形原语。注意对于一特定图形设备,并非所有图形原语都可以使用。这一点通过本章后面将原语和其他图形协议一起使用可以看得更加明显。

Color Mix 原语基于 RGB(Red Green Blue)颜色模型,表 2.2 所述的 IBM IRGB(Intensity Red Green Blue)颜色标准采用 4 比特,分别表示强度、红、绿、兰。在很多设备中,前 8 种标准颜色(0～7)都是通过三种基本颜色混叠形成的。另 8 种颜色(8～15)通过加一个强度比特生成,即当该比特为 1 时,产生的效果就像加入更多的白色墨水。

表 2.1　基本图形原语

原　语	描　述
Initialize	初始化图形设备
Draw	用当前画笔从当前点画到指定点
Move	从当前点移到指定点
Select Pen	选择指定编号的画笔(用于颜色控制)
Line Width	以像素点数为单位设置当前线宽
Line Style	从几种预定义线型选一种作为当前绘图格式
Foreground Color	从几种基于 IBMRGB 颜色标准的预定义混色方式中选一种
Bell	在图形设备上发出声响

表 2.2　IBM IRGB 颜色标准

混合编号	IRGB	混合名称
0	0000	Black
1	0001	Blue
2	0010	Green
3	0011	Cyan
4	0100	Red
5	0101	Magenta
6	0110	Brown
7	0111	White(Light gray)
8	1000	Dark gray
9	1001	Light blue
10	1010	Light green
11	1011	Light cyan
12	1100	Light red
13	1101	Light magenta
14	1110	Yellow
15	1111	Intense white

2.5　直接图形设备支持

现在已有相当多的图形语言能完成实际上相同的一些通用功能，这些语言能使对图形特