比特	色彩
000	黑
001	蓝
010	绿
011	青
100	红
101	品红
110	黄
111	白

这种方案可能只适合简单的类似卡通画的图像。真实世界出现的几乎所有颜色都是由红、绿、蓝三原色的不同色阶（levels）组合而成的。如果为每个像素赋予 2 个字节的存储空间，这样一来，可以给每一个原色分配 5 位（1 位保留）存储空间，这种方法可以表示出红、绿、蓝三种颜色且每种颜色具备 32 种不同的色阶，这样算下来总共有 32,768 种不同的颜色。这种模式通常称做高彩色（high color）或数千种颜色（thousands of colors）。

我们下面尝试一下用 3 个字节来表示一个像素，三原色中的每一种各占一个字节。这种编码模式使红、绿、蓝各自呈现出 256 种不同的色阶，这样算下来共有 16,777,216 种不同的颜色，这种方案通常叫做全彩色（full color）或百万种颜色（millions of colors）。如果视频显示器的分辨率为 640×480，即水平 640 像素，垂直 480 像素，将像素点的数量乘以表示每个像素点需要的字节数可以得到，共需要 921,600 字节的存储容量，即将近 1MB。

每个像素所赋予的比特数有时也称做色深（color depth）或色彩分辨率（color resolution）。颜色数与单个像素被赋予的比特数的关系如下：

$$\text{颜色数} = 2^{\text{每个像素所赋予的比特数}}$$

如果视频适配卡配备的存储器容量有限，那么它的最大色深或色彩分辨率自然而然也受到约束。假设有一个配备了 1 MB 存储器的视频适配卡，在每个像素被赋予 3 个字节的情况下分辨率可以达到 640×480。如果想把分辨率提高到 800×600，存储器就不足以为每个像素赋予 3 个字节，必须缩减到用 2 个字节来表示一个像素。

虽然现在来看在显示器上使用光栅技术似乎是很自然的事情，但是在早期，这种做法并不可行，因为在当时看来，这种技术需要的存储器空间太大。在这种情况下 SAGE 视频显示器应运而生，它是一种矢量（vector）显示器，相比电视机，它更像一种示波器。

电子枪可以通过电驱动定位到显示器任何一个像素点上，之后可以直接画出直线或曲线。由于屏幕上的图像具有持久性，不会立即消失，这样就可以利用直线和曲线形成最基本的画面。

支持光笔（light pen）是 SAGE 计算机的一大特色，操作者使用光笔可以改变显示器上的图像。光笔这种设备很特殊，从外观上来看是一端连有电线的笔。如果使用与之配套的软件，计算机能够感知到光笔所指的屏幕位置，随即根据光笔的位移相应地改变图像。

光笔的工作原理是什么呢？如果是第一次看到它，即使是相关领域的技术专家，也会感到困惑。理解它的关键在于光笔并不发射（emit）光——它所做的是检测（detect）光。对于 CRT（无论采用的是光栅还是向量显示技术），电子枪移动控制电路有两个最重要的功能，第一个功能是光笔一旦感知到电子枪射出的光，系统需要立即做出反应；第二个功能是系统在对其做出反应的过程中，需要确定出光笔指向的屏幕位置。

伊凡·苏泽兰（Van Sutherland，生于 1938 年）是最早预见到了计算机发展的一个全新领域，即交互式计算的人之一。在 1963 年，他演示了为 SAGE 计算机专门开发的名为“画板”（sketchpad）的程序。画板不仅可以将图像信息存放在存储器中，还可以把图像在屏幕上显示出来。你还可以使用光笔在显示器上画出图像并进行修改，与此同时，计算机会对光笔的轨迹一直进行跟踪。

还有一位早期交互式计算的预言家，那就是道格拉斯·恩格尔巴特（Douglas Engelbart，生于 1925 年）。他曾阅读过 1945 年万·布什发表的文章《思维之际》，巧合的是，五年之后他开始致力于研究计算机界面显示的新方法，并为之奉献毕生精力。20 世纪 60 年代中期，当恩格尔巴特在斯坦福研究所（Stanford Research Institute）工作时，他重新思考并设计了输入设备，提出了用五股（five-pronged）键盘作为指令输入设备（这个设备并未普及），另外还提出了一种配备轮子和按钮的设备，它的名字就是鼠标（mouse）。鼠标现在已经在全世界被广泛接受，它可以用来移动屏幕内的指针，还可以选择屏幕上出现的对象。

许多在早期热衷于交互式图形计算的科学家（但这里并不包括恩格尔巴特），他们不约而同地聚集在了施乐（Xerox）公司，幸运的是，此时的光栅显示器已经变得经济实用。施乐公司在 1970 年建立了帕洛阿尔托研究中心（Palo Alto Research Center，PARC），中

心的主要任务之一就是协助产品开发，以此来加快公司迈入计算机产业的步伐。PARC 中最著名的预言家应该算是阿伦·凯（Alan Kay，生于 1940 年），14 岁那年，阿伦·凯在一篇罗伯特·海因莱因（Robert Heinlein）撰写的故事中，读到了万·布什提出的微缩胶片图书馆，阿伦·凯因此而深受启发，不久他构想了一种名为“Dynabook”的便携式计算机。

PARC 着手的第一个大的工程是阿尔托（Alto），它的设计和制造完成于 1972-1973 年。从那个年代的标准去看，它是一个令人眼前一亮的产品。它采用落地式系统单元，配备 16 位处理器、2 个 3 MB 的磁盘驱动器、128 KB 的内存（最多可扩充到 512 KB），还包括一个三按钮的鼠标。在 Alto 开发的时候，16 位单芯片微处理器还未面世，所以它的处理器由将近 200 个集成电路组成。

Alto 有许多与众不同的地方，视频显示器是其中一个方面。屏幕的大小和形状就像一张纸——8 英寸宽，10 英寸高。它采用光栅成像技术，水平像素值为 606，垂直像素值为 808，算下来共有 489,648 个像素。其中每个像素占据 1 位存储空间，即每个像素取值只有两种：黑色或白色。视频显示的专用存储器容量为 64 KB，占用处理器的地址空间。

通过直接对视频显示存储器进行写操作，软件可以在屏幕上绘图或将不同字体、不同大小的文本显示在屏幕上。用户可以通过移动鼠标，在屏幕上对指针进行定位，还可以与屏幕上的对象进行交互。视频显示器与电传打字机在很多方面不尽相同，电传打字机顺序响应用户输入并按行将程序输出，而视频显示器的屏幕可以看做二维空间上的高密度的信息阵列，它还可以作为直接的用户输入源。

20 世纪 70 年代晚期，Alto 所搭配的程序逐渐凸显出很多新奇有趣的特点。比如窗口中可以容纳多个程序并同时显示在屏幕上。Alto 的视频图像功能使得软件从文本的束缚中摆脱出来，使其可以更加真实地反映用户的想法。图形对象（Graphical objects，比如按钮、菜单，以及被称做图标的小图片）成为用户接口的一员。鼠标可以在多个窗口中进行选择、触发图形对象来执行程序功能。

软件的内涵就在于此，它的意义远不止仅有的用户接口，还包括与用户的亲密耦合。软件使得计算机所涵盖的应用领域变得更广，而不仅仅局限于简单的数字变换。软件之所以被设计出来，其最终目的是——引用道格拉斯·恩格尔巴特在 1963 发表的一篇著名论文的标题——《为了扩展人类的智慧》（*For the Augmentation of Man's Intellect*）。

PARC 在 Alto 这个项目的开发成果预示着图形用户界面（Graphic User Interface，GUI）登上了历史的舞台。施乐公司并没有将 Alto 推向市场（价格定位 3 万美元以上绰绰有余）。从 10 年之后的今天来看，当时的 Alto 应该被包装成一种成功的消费产品并推向市场。

1979 年，斯蒂夫·乔布斯（Steve Jobs）带领苹果公司代表团对 PARC 进行了访问，在那里的所见所闻给他们留下了深刻的印象。而他们却花费了三年多的时间才推出具有图形界面的计算机，这就是在 1983 年 1 月推出的苹果莉萨（Apple Lisa），可惜这套系统在当时并不被看好。而一年以后推出的麦金托什机（Macintosh）却大获成功。

最原始的 Macintosh 机配备有 Motorola 68000 微处理器、64 KB 的只读存储器、128 KB 的随机访问存储器、一个 3.5 英寸的磁盘驱动器（存储容量为 400 KB）、一个键盘、一个鼠标和一个视频显示器，显示器水平像素为 512，垂直像素为 342（仅为 9 英寸的 CRT 对角线长度），像素总量为 175,104 个。每个像素赋予 1 位内存，只能显示黑白两色，这种配置约占 22 KB 的视频显示存储器。

最原始 Macintosh 机硬件方面很精巧，但是可更新能力很差。1984 年 Macintosh 操作系统的诞生对于 Mac（即 Macintosh 机）意义非凡，它的出现使得 Mac 变得与众不同，当时我们把这样一个操作系统称为系统软件（System Software），它就是现在著名的苹果操作系统（Mac OS）。

基于文本的单用户操作系统，如 CP/M 或 MS-DOS，体积很小但是不支持扩展的应用程序接口（API）。关于这点在第 22 章进行过解释，在这些基于文本的操作系统中，没有为访问文件系统的应用程序提供一种渠道。Mac OS 这种图形化操作系统所占的空间比前面提到的这两种要大得多，其中包含了上百个 API 函数，每一个函数都用其功能来命名。

MS-DOS 操作系统是基于文本的，如果要在屏幕上以电传打字机方式将文本显示出来，使用几个简单的 API 函数即可，但对于 Mac OS 这种基于图形的操作系统，必须提供一种在屏幕上显示图像的途径，程序通过这条途径对图像进行显示。从理论上来讲，一个 API 函数完全可以胜任这项任务，函数的功能就是设置某个水平和垂直坐标下的像素的颜色。但在实际应用中，这种方法效率较低以至于严重影响到了图像显示的速度。

在这种需求下，如果操作系统可以提供一整套图形编程系统，那么其意义是重大的，这样的操作系统必须包含如下 API 函数：画线、画矩形、画椭圆（包括圆）以及画出文

本。其中，线条可以是实线，可以是虚线，还可以是点线；矩形和椭圆可以具备不同的填充模式；字符可以具备不同字体和大小，还可以具备不同特效，如加粗和下画线等。图形编程系统负责规划如何将各式各样的图形对象以点阵集合的形式表示在显示器上。

如果程序在图形操作系统下运行，那么它们在显示器或打印机上画图这一过程中，使用的是一套完全相同的 API。正因为如此，字处理程序在屏幕上显示出的文档，与打印出来而得到的纸质文档，看上去非常相似。这种特点称为“所见即所得”（简写为 WYSIWYG）。这是喜剧演员弗雷普·威尔森（Flip Wilson）在扮演杰拉尔丁（Geraldine）角色中的一句话，这句话也成了计算机领域的一个经典口号。

图形用户界面对用户而言是极具吸引力的，其中一个重要原因就是不同的应用程序使用着大致相同的工作原理，并且影响着用户的使用经验。这样一来操作系统就承担起了支持 API 函数的重任，而应用程序就可以利用这些 API 函数去实现用户界面的不同组件，如按钮和菜单等。GUI 不仅是一种看上去简洁友好的用户环境，对于程序员而言，它还是一种重要的开发环境。程序员在开发新一代用户界面的时候可以不用从底层开始重新编写。

其实早在 Macintosh 问世之前，一些公司已经开始着手为 IBM PC 及其兼容机创建图形操作系统。这两种工作有一个显著的不同：苹果公司的硬件和软件都是由苹果公司自己设计的，因此开发人员的工作更加轻松。Macintosh 系统软件只支持一种类型的磁盘驱动器、一种视频显示器，以及两种型号的打印机。而 IBM PC 的图形操作系统开发人员所面对的是许多不同的硬件，操作系统与不同的硬件之间需要同时兼容。

还有一点，虽然 IBM PC 问世的时间（1981 年）较早，但 MS-DOS 应用程序已经在多数人心中根深蒂固，人们不愿意放弃它们。因此 PC 的图形操作系统必须具备一个重要的特性，那就是新的操作系统应该可以直接兼容 MS-DOS 应用程序，就好像 MS-DOS 应用程序是为新的操作系统专门设计的（Macintosh 不兼容 Apple II 系列软件，因为它们的微处理器型号不同）。

在 1985 年，迪吉多科研公司（Digital Research，CP/M 的后续公司）推出了图形环境管理器（Graphical Environment Manager，GEM）；VisiCorp（推出 Visilalc 软件的公司）推出了 VisiOn；与此同时微软公司发布了 Windows 1.0 版本，作为一匹黑马，当时它也被很多人认为将会成为“视窗争夺战”的胜利者。然而直到 1990 年 3 月 Windows 3.0 发布，

Windows 才真正受到大众瞩目。星星之火从那时开始燎原。在本书出版的 2000 年，约 90% 的小型计算机上使用的都是 Windows 操作系统。除了外观上的不同，Macintosh 和 Windows 这两种操作系统所包含的 API 也有着天壤之别。

从原理上来分析，除了图形显示器，图形操作系统与文本操作系统相比，对硬件支持的要求并没有太多不同。从理论上来讲甚至硬盘驱动器都可以算是多余的：比如最初的 Macintosh 没有配备，Windows 1.0 也不需要。虽然大家都认为使用鼠标操作会更加方便，但其实 Windows 1.0 可以不需要鼠标。

有一点很容易想到，随着微处理器速度越来越快，内存和外存的容量越来越大，图形用户界面将更加深入人心。图形操作系统将会支持越来越多的特性，它们所占的存储空间也将越来越大。2000 年左右的主流图形操作系统通常需要 200 MB 的硬盘空间和 32 MB 以上的内存。

在图形操作系统中，应用程序几乎都不使用汇编语言来开发。就拿早期的几款操作系统来看，Pascal 是 Macintosh 下的主流开发语言。在 Windows 操作系统中，C 语言一统江湖。还有一个不得不提到的例子，那就是 PARC 向我们展示的一种全新的方法。大概从 1972 年开始，PARC 的研究员着手开始研发一种名为 Smalltalk 的语言，这种语言嵌入了面向对象程序设计思想（Object-Oriented Programming），也就是今天的 OOP。

从传统意义上来讲，高级程序设计语言会自然而然地区分出代码（比如以 set、for、if 这样的关键词开头的语句）和数据，即变量所代表的数字。这种区分毫无疑问来自于冯·诺依曼计算机的体系结构。在这样一种体系结构中，只有两种元素，一种是机器码，一种是机器码所操作的数据。

在面向对象的程序设计中，和冯·诺依曼计算机的体系结构所不同的是，对象（object）实际上是代码和数据的组合。在对象内部，与其相关联的代码决定了数据存在的意义，要理解数据的存储方式首先需要理解代码。对象如果需要与其他对象通信，则通过发送或接收消息（message）来实现这一过程，比如一个对象可以通过给另一个对象发送指令来获得相应信息。

在图形操作系统的应用程序开发过程中，面向对象语言可以算得上是一种很不错的工具，因为编程人员处理屏幕上的对象（如窗口和按钮等）的过程就是用户感知屏幕元

素的过程。举例来讲，假设按钮是面向对象语言中的一个对象。屏幕上的按钮具备一定尺寸和位置，按钮上可以显示文本或小图标，这些都可以抽象成为与对象关联的数据。如果用户通过键盘或鼠标按下按钮，系统就会向按钮对象发送一个表示其被触发的消息，该按钮对象收到消息后就会调用与自身关联的代码进行响应。

小型计算机上最流行的面向对象语言是一种对传统的类似于 ALGOL 语言的扩展，C 和 Pascal 就属于此类。由 C 扩展的面向对象语言就是赫赫有名的 C++（我们可以回忆一下，两个加号放在一起等价于 C 语言中的自增操作）。C++的核心思想大部分来自于贝尔电话实验室（Bell Telephone Laboratories）的贾尼·斯特劳斯特卢普（Bjarne Stroustrup，生于 1950 年），最开始 C++是作为一种转换程序，它可以把编写的程序转换成 C 程序（但是转换出的C程序即难看又难以理解）。转换完成之后的C程序可以像普通程序一样编译。

其实，面向对象语言能做到的，传统语言也能做到。但是编程终究是人类发明的一种解决问题的活动，面向对象语言使得编程人员多了一种可选的解决方案，这种解决方案具备更加优越的组织结构。如果你想——虽然困难重重——面向对象语言编写的一种程序，并使其在 Macintosh 和 Windows 上都可以编译后运行，这是完全可以做到的。此类程序并不直接引用 API，使用的是被称为 API 函数的对象。Macintosh 和 Windows 使用两种不同的对象定义来编译程序。

许多在小型计算机上工作的编程人员已经逐渐不用命令行编译程序，而是使用集成开发环境（Integrated Development Environment，IDE）。这个环境里集成了所有需要的工具，而环境本身可以像其他图形应用程序一样运行，这样一来就大大简化了程序开发任务。还有一种称做可视化编程（Visual Programming）的技术被程序开发人员广泛利用，按钮及其他组件可以通过鼠标拖曳进行“排版”，从而达到在窗口交互设计的目的。

在第 22 章中我们一起讨论过文本文件。为了方便人们阅读，这类文件仅由 ASCII 字符组成。我们回想一下使用基于文本的操作系统的那个年代，文本文件是应用程序之间进行交流的理想媒介。它的最大优点是可检索性——程序可以检索多个文本文件，然后确定它们中是否有文件包含某一字符串。但如果操作系统中有一种机制用来显示不同字体、大小，以及不同效果比如斜体、黑体和下画线，那么文本文件就不再适用了。很多字处理软件其实都会使用一种自己独有的二进制格式来存储文档。文本文件同样也不适用于图形信息。

但我们要清楚的是，与文本相关的信息（比如，字体及段落版式），都可以被编码，而且编码后并不影响其可读性。这种方案的关键是选用一个适当的转换字符来标识出这些信息。在 Microsoft 设计的富文本文件格式（rich text format，RTF）中，大括号“{”和“}”以及反斜杠“\”封装了文本的格式信息，RTF 也成为了应用程序间传递格式化文本的一种方法。

PostScript 作为一种文本格式，将这种概念发挥到了极致。PostScript 的设计者是 Adobe 系统的创始人之一——约翰·沃诺克（John Warnock，生于 1940 年）。PostScript 是一种通用的图形编程语言，在 2000 年时主要用在高端计算机的打印机上，用于显示字符或图形。

随着硬件性能逐渐提升，价格日渐便宜，图形显示与个人计算环境的融合已是大势所趋。微处理器的处理速度越来越快，存储器价格越来越低廉，视频显示器及打印机分辨率不断增加，而且支持的颜色数目也成千上万，这一切大大推动了计算机图形界发展。

计算机图形也逐步产生了两种分支——本章的前面曾提到过这两个词，当时是为了区分图形视频显示器——这两个分支就是矢量和光栅。

矢量图形（vector graphics）在一些算法的帮助下，利用直线、曲线及填充区域生成图形。这也正是计算机辅助设计（Computer-Assisted Drawing，CAD）所应用的领域。矢量图形在工程和体系结构设计中有着十分重要的作用。矢量图形一般转化为图元文件（metafile）格式以存放到文件中。图元文件是由生成矢量图形的一系列绘制命令的集合组成的，这些命令通常已经被编码为二进制形式。

矢量图形的主要工具就是直线、曲线及填充区域。如果你想设计桥梁，使用矢量图形来实现将很简单，但如果要显示桥梁的实际结构，矢量图形就显得无能为力了。对于现实世界里的一副桥梁的整体结构图，用矢量图形来表示将会很复杂，而且困难重重。

光栅图形（也称做位图），就是为了解决这一问题应运而生的。位图（bitmap）将图像以矩阵阵列的形式进行编码，阵列中的一个单位对应着输出设备上的一个像素点。就像视频显示器一样，位图是一种空间上的概念（可以称其具有分辨率），其图像的宽度和高度都以像素为单位来表示。位图也具备色深（也可叫做颜色分辨率/颜色深度）的概念，色深是指每一个像素被赋予的比特数。位图中每个像素被赋予的比特数相同。

尽管位图从表现形式上看是二维的，但其本身的存储形式却是一串连续的字节——通

常从最顶端 1 行像素开始，紧跟着的是第 2 行、第 3 行，等等。

有些位图产生于图形操作系统设计的绘制程序，它们都是由某个操作者利用这些程序“手工绘制”出来的，还有一些位图是由计算机代码通过某种算法产生的。如今很多现实中的场景都利用位图来表示（比如数码照片），要把现实世界的图像输入到计算机中，可以借助一些不同的硬件，这类设备一般统称为电荷耦合器（charge-coupled device，CCD），它是一种在光照下会起电的半导体器件。每个像素都需要一个 CCD 单元来进行采样。

这些设备中最原始的可以算是扫描仪（scanner）了，其原理和影印机类似，都是利用一行 CCD 扫过需要复印的图像的表面，比如照片。由于光感度不同，不同区域 CCD 累积的电荷数也不同。扫描仪的配套软件把图像转换成位图存放在文件中。

视频摄像机利用二维 CCD 单元阵列捕捉图像。通常被捕捉到的图像存放在录像磁带上。但其实视频输出可以直接交给视频帧采集器（video frame grabber）去处理，它的工作原理是把模拟信号转换成像素值阵列。帧采集器可以收集通用的视频源，如录像机（Video Cassette Recorder，VCR）或激光影碟机（Laser Disc Player），甚至可以用于有线电视机机顶盒。

近几年，数码相机的价格已逐渐降低到了家庭用户可以承担的水平，它们从外观上来看和一般相机几乎一样。但数码相机并不使用胶片，而是使用 CCD 阵列来捕捉图像并直接将其转移到相机的存储器中，之后在适当的时候转储至计算机中。

图形操作系统通常可以将位图文件以某种格式进行存储。Macintosh 系统采用 Paint 格式，之所以叫这个名字是参考了创建这种格式的 MacPaint 程序（Macintosh 的 PICT 格式同时支持位图和矢量图形，而且是它们的首选格式）。Windows 里的默认的格式是 BMP，位图文件通常以它作为扩展名。

位图文件可能很大，如果有方法可以让它们变小一些那就再好不过了。这种需求催生了计算机科学中的数据压缩（Data Compression）这一全新领域。

假设我们正在处理一幅每个像素占 3 位的图像，这种图像在本章前面曾讲过。这张图片上出现的画面是一片天空、一栋房子和一块草坪。因此，图片中可能有大片的蓝色和绿色。很可能位图的最上面一行出现了 72 个蓝色像素。如果有一种方法可以表示蓝色

像素连续且重复了 72 次，那么通过这种方法表示的位图文件将会比原先的小很多。这样的压缩方法称为游程长度编码（Run-Length Encoding），即 RLE。

通常办公室的传真机采用的就是 RLE 压缩方法，压缩过程一般在传真机通过电话线传送图像之前。由于传真机展现出的图片都是黑白两色，没有灰度和彩色，所以通常像素值都会有很长串的白色区域，适合使用 RLE 压缩。

这十多年里风光无限的位图文件格式是图形交换格式（Graphics Interchange Format）即 GIF，由计算服务（CompuServe）公司于 1987 年开发。GIF 文件所采用的压缩技术称为 LZW，LZW 源自其三位创建者的名字：Lemplel、Ziv 和 Welch。LZW 比 RLE 更加强大，因为它所考虑的是像素值的模式（patterns），而 RLE 针对的是具有相同值的像素串。

RLE 和 LZW 都属无损（lossless）压缩技术范畴，因为可以从压缩数据中重新生成完整的初始文件。专业一点的说法是，压缩过程是可逆的（reversible）。可逆压缩方法并不适用于所有类型的文件，这点不难证明。在某些情况下，采用这些方法“压缩”后的文件比初始文件还要大！

近几年来看，有损（lossy）压缩技术大行其道。有损失的压缩是不可逆的，这是由于部分原始数据在压缩过程中被丢弃了。有损压缩技术不应该用于电子报表或文字处理文档，因为在这些重要文档里面少一个数字或者字母都会“失之毫厘，谬以千里”。但对于压缩图像，这些损失还是可以接受的，部分数据的损失不会使图片的整体效果有太大的变化。这就是为什么有损压缩技术的思想起源于心理视觉的原因，心理视觉领域所探究的是人的视觉，并根据心理因素确定人们所看到的景象中哪些比较重要而哪些不重要。

在 JPEG 中，人们使用了一系列具有重大意义的位图有损压缩技术，。JPEG 代表的是联合图像专家组（Joint Photography Experts Group），它涵盖了几种压缩技术，其中一些是无损的，另一些是有损的。

把图元文件转换成位图文件的方法很简单。这是由于视频显示存储器与位图在概念上保持一致。如果程序能把图元文件画在视频显示存储器中，则它也能在位图上画出图元文件。

但从位图文件到图元文件的转换却不那么容易，如果位图文件过于复杂甚至会导致无法转换。为了解决此类问题，人们发明了一项技术，那就是光学字符识别（Optical

Character Recognition），或简称为 OCR。对于位图上出现的一些字符（来自于传真机，或来自于页面扫描），如果需要转换成 ASCII 码，那么 OCR 就会派上用场。OCR 软件会对比特流进行模式识别，然后确定其代表的字符。由于算法的复杂性，OCR 软件并不能保证百分之百准确。虽然不很准确，但是 OCR 软件一直尝试将手写的字符也转换成 ASCII 码字符。

位图和图元文件都是数字化的可视信息。同理，音频信息也能转换成比特和字节。

随着 1983 年激光唱片（compact disc）的问世，数字化音响掀起了一轮消费狂潮，它同时也成为了最成功的电子消费品案例。CD 是由飞利浦和索尼公司联合开发出的产品，一张直径为 12 cm 的 CD 可存储 74 分钟的数字化声音。而这个 74 分钟的时长因为贝多芬的第九交响曲（Beethoven's Ninth Symphony）刚好可以存储在一张 CD 上。

CD 中声音信息采用的编码技术被称为脉冲编码调制技术（Pulse Code Modulation），简称为 PCM。名字听起来有点故弄玄虚，但从概念上讲 PCM 其实是一种很简单的过程。

振动是声音之源。人们发出的声音来源于声带的振动，大号的声音也来自于振动，森林里的树倒下的声音归根结底也是振动，还有很多例子，它们的振动来源于空气分子的移动。空气在被推拉的过程中，有时压缩有时放松，有时向后有时向前，这个过程每秒钟可以达到成百上千次，最终使耳膜产生振动，从而我们能够听到声音。

声波可以被模拟，一个成功的例子就是 1877 年爱迪生发明的第一台电唱机，它利用锡箔圆桶表面上的隆起和凹陷部位录制和播放背景音乐。直到 CD 的出现，这种录制技术才发生了稍许变化，圆桶变成了光盘，锡箔变成了塑料。早期的电唱机是机械化的，但是后来，电子放大器被用来放大声音。声音可以通过麦克风上配备的可变电阻转换成电流，喇叭中的电磁铁又可以将电流转换为声音。

代表声音的电流与先前所讲过信号不同，本书之前讨论过的是在“连通——断开”之间跳变的 1/0 数字信号。声波的变化是连续的，因而产生电流的电压也是如此。在这里电流产生的目的是模拟（analog）声波。为了达到这个目的，我们使用了一种新设备，通常把它叫做模拟数字转换器（Analog To Digital Converter，ADC）——所有的功能集成在了一个芯片上——将模拟电压转换成二进制数表示。一定长度的数字信号将会被 ADC 所输出——通常长度为 8、12 或 16 个比特——它们组合在一起表明了电压的相对级别。如果

ADC 的转化长度为 12 比特，那么电压值的取值范围为 000h～FFFh，这样一来就可以区分出 4096 个不同的电压级别。

在一种名为脉冲编码调制（Pulse Code Modulation）的技术中，以电压形式表示的声波将以恒定的频率被转换成数值。而这些数值将以小孔的形式刻在光盘表面，通过这种方式，电压就以数值的形式被存储在 CD 上。要读取这些信息时，可以通过分析从 CD 表面反射的激光读取到所存储的数值。在播放声音的时候这些数值又被转换成电流，这一过程利用到了数字模拟转换器（Digital-To-Analog converter，即 DAC，DAC 还可以用在彩色图形板上，将像素值转换成模拟信号并传输至显示器）。

声波电压在恒定的频率下被转换成了数字，该频率被称为采样率（sampling rate）。1928 年，贝尔电话实验室的哈里·奈奎斯特（Harry Nyquist）通过证明得到了一个总要结论：采样频率应至少为被采样信号（即被记录和播放的信号）最大频率的两倍。人类可以听到的声音的频率范围通常为 20～20,000Hz。CD 使用的采样频率为每秒 44,100 次，比人类听觉范围最大频率的两倍还要大一些。

CD 中存储的声音的动态变化范围决定了每次采样的比特数，这个范围就是 CD 存储声音的最高与最低频率之差。这难免给人感觉有些复杂：电流通过不断变化来模拟声波，其达到的最高峰称为声波的振幅（amplitude）。我们感知到的声音强度是振幅的两倍。1 贝尔（bel，这个名称来源于 Alexander Graham Bell 最后一个单词的三个字母），代表着强度的 10 倍；1 分贝（decibel）代表着 1 贝尔的十分之一。1 分贝代表着人类所能感知到的声音最小强度变化。

如果每次采样大小为 16 比特，这样就能够表示 96 分贝的动态范围，这一范围的下限是刚好能听到的声音的阈值（低于这一值的声音是听不到的），而上限就是人们承受最大负荷声音的极限阈值。CD 光盘的每个采样点就是用 16 比特表示的。

综上所述，CD 光盘中每秒产生 44,100 个采样样本，每个样本占据 2 个字节。有时你还希望享受立体声效果，这样的话采样信息需要翻倍，则每秒总共需要 176,400 字节，算下来每分种需要 10,584,000 个字节（这是个庞大的数字，正因如此，20 世纪 80 年代前数字记录声音的方法并没有普及）。74 分钟的立体声 CD 需要字节数为 783,216,000。

相对于模拟声音，数字化声音的优点不言而喻。特别是在模拟声音被复制的时候（例

如把录音磁带转录成电唱片）难免会有些失真。而对于用数字形式表示的数字化声音而言，都可以实现无失真的转录或复制。细心观察的人会发现，过去在电话通话中，信号传输线路越长则声音越糟。这种情况已经一去不复返了，因为现在大部分电话系统都已经数字化了，即使你打跨国电话，听筒中的声音也像在对街对话一样清晰。

CD 可以存储声音，也可以存储数据。专门用来存储数据的 CD 统称为 CD-ROM（CD 只读存储器），通常 CD-ROM 最大存储容量约为 660 MB。如今许多计算机中都配备 CD 驱动器，而应用程序及游戏软件都可以存储在 CD-ROM 中。

声音、音乐、视频逐渐走入个人计算机是在大约 10 年前，当时统称为多媒体（multimedia），虽然这几年发展很迅速，但这个名字已经普遍使用开来，所以也就不需要新起一个特别的名称了。如今的家用计算机都配备声卡，声卡中包含一个 ADC，它可以将麦克风传输的模拟声音信号转储成为数字信号，此外还包括一个 DAC，它的作用是帮助扩音器播放录制的数字声音。声音还可以按照波形文件（waveform files）方式存放于磁盘。

在我们使用家用计算机录制和播放声音的时候，其实对声音质量要求并不高，一般不会苛求达到 CD 的效果，所以 Macintosh 和 Windows 这两种操作系统提供了较低的采样频率，尤其是 22,050 Hz、11,025 Hz 和 8,000 Hz 这三种频率，采样信息量也保持在较少的 8 位，并且使用了频度录制手段。声音录制时所占的存储容量也减少到了每秒 8000 字节，这样算下来每分钟约占 480,000 字节。

很多人在科幻电影及电视剧集中看到过这样的场景，未来的计算机正在用纯正的英语与用户进行交互。只要计算机配备了录制和播放数字化声音的硬件，那么实现这个目标不过是一个软件问题。

如果要使计算机能够用易于理解的单词和句子与人交谈，有很多种方法。一种方法是预先录制句子、短语、单词还有数字，然后将它们以文件的形式存储，之后使用多种形式将其糅合在一起。这种方法常见于电话公司的信息系统中，在这种环境下只需要有限的单词、数字及其组合，因此这种方式能适用。

有一种更加通用的声音合成方法，它把 ASCII 码字符转换成波形数据。例如在英语拼写方面，由于存在很多不一致性，软件系统需要使用一个词典或复杂的算法来确定单

词的准确发音。一个完整的单词由基本的音节（也叫做音素，phonemes）组成。一般情况下软件都需要做一些其他方面的调整。如果一个句子后紧跟着的是问号，则最后一个单词的发音频率必须相应提高。

语音识别（voice recognition）——把波形数据转换成 ASCII 码字符——这个是一个极其复杂的过程。其实许多人在理解口语化的方言时也有很多困难。在使用个人计算中的语音处理软件时，通常需要对软件进行样本训练，以便软件能尽量准确地转录某个人的话语。这中间涉及的问题已经大大超出了 ASCII 码文本转换的范畴，从本质上来讲是一个利用编程技术使得计算机“理解”人类语言的过程。这类问题正是人工智能（artificial Intelligence）所研究的领域。

当今计算机中的声卡还配备了小型电子音乐合成器，它能模仿 128 种不同的乐器和 47 种不同的打击乐器，这类设备被称做 MIDI 合成器。MIDI 即乐器数字接口（Musical Instrument Digital Interface），这项发明出现于在 20 世纪 80 年代早期，由一家电子音乐合成器制造协会推出，主要用于将电子乐器组合起来，并且连到计算机上。

MIDI 合成器的类型不同，它们合成乐器声音的方法也就不同，有一些方法的效果更加逼真。MIDI 合成器所展现出来的特性已经超越了 MIDI 所定义的范畴。但其实它的功能就是以演奏声音的方式来响应短消息序列——长度通常为 1、2 或 3 个字节。MIDI 消息本身就指明了所需要的乐器、将要演奏的音符，或正在演奏的音乐的休止符。

MIDI 文件不仅包含了一系列 MIDI 消息集合，还包括了时间信息。通常一个 MIDI 文件“麻雀虽小，五脏俱全”，包含了整套演奏信息，因此可以由计算机上的 MIDI 合成器完整地演奏。相比于包含同样一段音乐的波形文件，MIDI 文件通常要小得多。就文件的相对大小而言，如果把一个波形文件比作位图文件，则 MIDI 文件就好像矢量图元文件。MIDI 文件的缺点来源于 MIDI 合成器的异构性：同样一个 MIDI 文件可能在一个合成器上完美演奏，但在另一个合成器上可能不堪入耳。

数字化电影是多媒体的另一片天地。把一系列静止图像快速播放，可以达到电影和电视图像中出现的物体移动效果。我们把这期间出现的单个图像称为帧（frames）。电影的播放速率为 24 帧/秒，北美的电视节目为 30 帧/秒，世界上大部分地方的电视为 25 帧/秒。

计算机中的电影文件一般都是由一系列附带声音的位图组合而成。但是如果不经过压缩处理，一部电影文件中的数据量将会很大。假如电影中每一帧包含的像素大小是640×480，每个像素为24位真彩色，那么每一帧的大小就为921,600字节。如果播放速度为30帧/秒，则每秒需要的存储空间为27,648,000字节。照这样计算下去，每分钟需要的空间大小为1,658,880,000字节，一部两小时的电影需要199,065,600,000字节，大约200 GB。正因如此，我们的计算机上播放的电影经常很短、解析度很小，而且清晰度很差。

就像JEPG压缩技术可以用来减少静态图像所占的数据空间一样，MPEG压缩技术用于处理动态电影文件。MPEG全称是移动图像专家小组（Moving Pictures Expert Group）。动态图像压缩技术基于的是一种客观事实，即每一帧继承了前一帧的大部分信息，也就是说存在冗余信息。

针对不同的多媒体产品有不同的MPEG标准。MPEG-2用于高清晰度电视（high-definition television，HDTV）及数字影音光盘（digital video discs，DVD）。DVD也叫数字多用光盘（digital versatile discs），其物理大小与CD一样，但是DVD的两面都可以记录数据而且每一面有两层。在DVD中，视频信息可以压缩为原始大小的五十分之一，因此前面提到的一部两小时的电影将占据4 GB的空间，而且只占据其中一面的一层。所以一张具有两面四层的DVD盘容量可达到16 GB，容量达到了CD的25倍。如果不出预料，在未来的某一天，DVD将替代CD-ROM成为新的软件存储媒介。

随着CD-ROM和DVD-ROM的出现，这是否意味着万·布什预言的麦克斯储存器在今天成为了现实？最原始的麦克斯储存器设想中，使用的原材料是缩微胶片，但显然CD-ROM和DVD-ROM更合适担此重任。由于电子媒体易于检索，它们比物理媒体更具有优越性。可惜同时访问多个CD或DVD驱动器并不现实。我们的存储设备有一点与万·布什所提出的概念有所不同，那就是所有信息并不一定要触手可及，真正使它们达到信息共享的做法是计算机互连，这样做还可以更有效地利用存储空间。

对计算机进行公开性远程操作的第一人是乔治·史帝比兹（George Stibitz），也正是他，在20世纪30年代设计了贝尔实验室的继电器计算机。对继电器计算机的远程操作的演示地点在达特茅斯，时间是在1940年。

电话系统在线路上传输的是声音，而不是比特。在电话线路上传输比特需要先将其转换成声音，传输完之后在转换回比特。单一频率和振幅的连续声波（统称为载波，carrier）