第25章　图形化革命

1945年9月10日，《Life》杂志的读者看到的大多是平常的一些文章和照片：有关第二次世界大战结束的故事，舞蹈家Vaslav Nijinsky在维也纳生活的报道，一则有关美国汽车工人的图片报道。但那一期的杂志也有意想不到的东西：一篇 Vannevar Bush（1890-1974）的关于科学研究的未来的展望性文章。 Van Bush（人们这样称呼他）在计算机历史上写下了重要的一笔。在1927年～1931年任麻省理工学院工程教授期间，他设计了一种具有重大意义的模拟计算机 —— 微分分析器。1945年，在《Life》杂志发表这篇文章的时候， Bush是科学研究和发展部的主管，负责协调美国在战争期间的科学活动，包括曼哈顿计划。

通过对两个月前第一次发表在《 The AtLantic Monthly》上的那篇文章的精简， Bush在《Life》杂志上的文章《As We May Think》描述了未来一些假想发明，希望科学家和研究人员解决日益增多的技术杂志及文章。 Bush谈到了作为一种解决方案的微缩胶片，勾划出了一种他称之为 Memex 的设备来保存书、文章、记录和书中的图片。 Memex也可根据人们所想到的关系在这些东西之间建立有关某个主题的联系。他甚至设想出了一个新的职业群体，他们可在大量的信息载体之间牢固地建立起联系。

尽管描绘未来光辉前景的文章在20世纪很普遍，但《As We May Think》则不同，它描述的既不是关于减轻家务负担的设备的故事，也不是关于未来交通运输或机器人的故事，而是关于信息及如何用新技术处理信息的故事。

从第一台继电器计算器制造出来已经历了 65年，计算机变得越来越小、越来越快，也越来越便宜。这种趋势已改变了计算机的本质。当计算机越来越便宜时，每一个人都可拥有自己的计算机。计算机越小、越快，软件则变得越高级，同时机器可以完成更多的工作。

更好地利用这些额外功能和速度的一种方法就是改进计算机系统中至关重要的部分，即用户接口 —— 人和计算机的交互点。人和计算机是差别很大的两种物质，但不幸的是，说服人们调整以适应计算机的特性是比其他方法更容易的方法。

早先，数字计算机根本上不是交互式的。有些使用开关和电缆编程，有些使用穿孔纸带或胶片编程。到20世纪50年代和60年代（甚至延续到70年代），计算机进化到使用批处理：程序和数据穿孔成卡片，然后读入到计算机内存，程序分析数据，得出一些结论，再在纸上打印出结果。

最早的交互式计算机使用的是电传打字机。如前一章讲到的 Dartmouth分时操作系统（始于20世纪60年代早期）支持多个电传打字机，可以同时使用。在这样的系统里，用户在打字机上敲一行，计算机以回答一行或多行作为响应。打字机和计算机之间的信息交流全部是ASCII码流（或其他字符集），除了像回车换行这样简单的控制码外，差不多全是字符代码。事务只是按纸卷的方向进行。

然而，阴极射线管（在 70年代就已经很普遍了）则没有这些限制。可以用软件以更灵活的方式来处理整个屏幕。然而，可能是为了设法保持显示器输出符合操作系统的显示输出逻辑，早先为小型计算机编写的软件不断地把 CRT显示器作为“ 玻璃打字机 ” —— 一行一行地显

示直到布满整个屏幕，当有字符到达屏幕底端时，屏幕的内容要向上翻滚。 CP/M和MS-DOS中的实用程序都是以电传打字机的模式来使用视频显示器。也许原型电传打字机的操作系统是UNIX，它仍然保持着这种传统。

令人感兴趣的是，ASCII码字符集并不都适用于阴极射线管显示。在最初设计 ASCII码时，代码1Bh标识为Escape，专门处理字符集的扩充。 1979年，ANSI印发了一个标准，题为“使用ASCII码的附加控制”。该标准的目的是“为了适应可预见的有关二维字符图像设备输入输出控制的要求，包括有阴极射线管和打印机在内的交互终端…”

当然，Escape的代码1Bh只有1个字节，且只有一个含义。 Escape通过作为可变长序列的开端来表达不同的功能。例如，以下这个序列：

```
1Bh    5Bh    32h    4Ah
```

即Escape 代码后面跟上字符[2J，定义成删除整个屏幕并移动光标至左上角。这是在电传打字机上所不能实现的。下面这个序列：

```
1Bh    5Bh    35h  3Bh    32h    39h    48h
```

即Escape代码后面跟上字符[5；29H，把光标移到第5行的第29列。

由键盘和CRT组合而成，对来自远方计算机的 ASCII码（也可能是Escape序列集合）作出响应，这样的设备有时称作哑终端。哑终端比打字机要快并且从某种意义上讲也更灵活，但是它并没有快到足以引起用户界面的真正创新。这种创新来自于 20世纪70年代的小计算机，正如第21章中的假想计算机，这种计算机有视频显示存储器作为微处理器地址空间的一部分。

家用计算机显著区别于它们大而昂贵的伙伴的第一个标志可能是 VisiCalc的使用。VisiCalc由Dan Bricklin (生于1951年)和Bob Frankston(生于1949年)设计和编程，于1979 年推出，用于Apple II。VisiCalc在屏幕上呈现给用户一个二维电子数据表。在 VisiCalc出现之前，报表通常是一张纸，使用行、列来进行一系列计算。 VisiCalc用视频显示器取代了纸，使得用户可以移动报表，输入数据或公式，在进行修改后重新计算每一项。

令人吃惊的是VisiCalc是不能复制到大型机上的应用程序。像 VisiCalc这样的程序需要很快地刷新屏幕。因此，它直接向 Apple II的视频显示器使用的随机访问存储器写入。该存储器是微处理器地址空间的一部分。大型分时计算机和哑终端之间的接口速度不是很快，从而使得电子报表程序不能使用。

计算机响应键盘、刷新视频显示器的速度越快，用户和计算机潜在的交互就越紧密。在IBM PC 机出现的头10年（20世纪80年代），为它编写的大多数软件是直接写入显示存储器的。由于IBM建立了一套硬件标准，其他计算机厂商追随这一标准，使得软件厂商可以绕过操作系统直接使用硬件而不用担心他们的软件在某些机器上不能正确运行（或根本不能运行）。如果所有的PC“克隆体”都与它们的视频显示器有不同的硬件接口，则对软件厂商来说要满足所有不同的硬件设计是非常困难的。

IBM PC所使用的早期的应用程序大多数只有字符输出而没有图形输出。使用字符输出同样能使得应用程序的执行速度加快。如果视频显示器设计得如第 21章所描述的那样，则程序只需简单地把某个字符的ASCII码写入内存就可以在屏幕上显示出该字符。使用图形视频显示的程序需要写入8个或更多的字节到内存中才能画出文本字符的图形。

从字符显示到图形显示的变化是计算机革命中极其重要的一步，然而图形方式下计算机硬件和软件的发展比文本和数字方式下计算机硬件和软件的发展要慢的多。早在 1945年，

冯·诺依曼就设想了一种像示波器一样的显示器，可用来使信息图形化。但是，直到 20世纪50年代早期，计算机图形才开始成为现实。当时麻省理工学院（得到 IBM资助）建立了林肯实验室来开发计算机，用于美国空军的空中防卫系统。该项目称为 SAGE（semi-automatic-ground environment）,有一个图形显示屏帮助操作员分析大量数据。

在SAGE这样的系统中使用的早期视频显示器不像今天我们在 PC机中所用的显示器。今天普通的PC机显示器是光栅显示器。就像电视机，所有的图像由一系列水平光栅线组成，由一个电子枪射击光束很快地前后移动扫过整个屏幕形成光栅。屏幕可以看成是一个大的矩形点阵，这些点称为像素。在计算机里，一块内存专门供视频显示使用，屏幕上的每一个像素由1位或多位表示。这些位值决定像素是否亮，是什么颜色。

例如，今天多数计算机显示器有至少水平方向 640像素的分辨率，垂直方向 480像素的分辨率，像素的总数是这两个数的乘积 307 200。如果1个像素只占用1位内存，则每个像素只局限于两种颜色，即通常的黑、白色。如， 0像素为黑，1像素为白。这样的视频显示器需要 307 200位的内存，即38 400字节。

随着可能的颜色数目的增多，每个像素需要更多的位，显示适配器也需要更多的内存。例如，每个像素可以用一个字节来编码灰度。按照这样的安排，字节 00h 为黑，FFh为白，之间的值代表不同的灰度。

CRT上的彩色由三个电子枪产生，每一个分别负责三原色红、绿、蓝中的一种（可以用放大镜来观察电视机或彩色计算机屏幕以验证它的正确性。由不同的原色组合来显示图像），红色和蓝色的组合是黄色，红色与绿色的组合是洋红色，蓝色和绿色的组合是青色，三原色的组合是白色。

最简单的彩色图像显示适配器每个像素需要 3位。像素可以如以下这样编码，每一个原色对应1位：

位	颜色
000	黑
001	蓝
010	绿
011	青
100	红
101	洋红
110	黄
111	白

但是，这种方式只适合于简单的类似卡通画的图像。许多现实世界中的颜色都是红、绿、蓝按不同级别组合而成的。如果用 2个字节来表示一个像素，则每一个原色可分配 5位（1位保留），这样可以给出红、绿、蓝三种颜色各 32种不同的级别，即总共可有 32 768种不同的颜色。这种模式通常称作高彩色或千种颜色。

下一步是用3个字节来表示一个像素，每种颜色占一个字节。这种编码模式使红、绿、蓝三种颜色各有256种不同的级别，这样总共有 16 777 216种不同的颜色，通常称作全彩色或百万种颜色。如果视频显示器的分辨率为水平 640像素，垂直480像素，则总共需要921 600字节的存储容量，即将近1M字节。

每个像素所占的位数有时也称作颜色深度或颜色分辨率。不同颜色数量与每个像素所占的位数的关系如下：

$$\text{颜色数} = 2^{\text{每个像素所占位数}}$$

视频适配卡只有一定数量的存储器，这样它所能达到的分辨率和颜色深度将受到限制。例如，一个具有1M字节存储器的视频适配卡可以达到640×480的分辨率，每个像素占3个字节。如果想用800×600的分辨率，则没有足够的存储器给每个像素分配3个字节，而要用2个字节来表示一个像素。

尽管现在使用光栅显示器似乎是理所当然的，但在早期由于需要大量存储器，使用光栅显示器就不太实际。SAGE视频显示器是矢量显示器，比电视机更像示波器。电子枪可以定位到显示器上任何部分的点，并且直接画出直线或曲线。利用屏幕上图像的可持续性使得能用这些直线和曲线来形成最基本的画面。

SAGE计算机也支持光笔，操作者可用来在显示器上改变图像。光笔是特殊的设备，看起来像一只铁笔，一端连有电线。运行适当的软件后，计算机可以检测到光笔指向的屏幕位置，并随着光笔的移动而改变图像。

光笔是如何工作的呢？即使是技术专家，在第一次看到光笔的时候也会迷惑不解。关键在于光笔不发射光——它检测光。在CRT中（无论是用光栅还是向量显示），控制电子枪移动的电路也可以确定从电子枪射出的光何时打到光笔上，从而确定光笔正指向屏幕的什么位置。

Van Sutherland(生于1938年)是预见到新的交互式计算时代的的多个人之一，他在1963年示范了一个他为SAGE计算机开发的名为Sketchpad（画板）的具有革命性意义的图形程序。画板可以存放图像信息到存储器，并且可以把图像显示在屏幕上。另外，还可以用光笔在显示器上画图并修改，同时计算机会一直跟踪它。

另外一个交互式计算的设想家是Douglas Engelbart(生于1925年)。他曾读过1945年Vannevar Bush发表的文章《As We May Think》，并在5年后开始用一生的时间致力于研究新的计算机界面。20世纪60年代中期，当他在Sanford研究院时，他彻底重新考虑了输入设备，提出了用有五个尖端的键盘(此设备没有流行)以及一个他叫作鼠标的有轮子和按钮的小设备来输入命令。鼠标现在已广泛用来移动屏幕上的指针，以选择屏幕上的对象。

恰逢光栅显示器在经济上切实可行，许多早期的热衷于交互式图形计算的人们（尽管没有Engelbart）也已聚集在Xerox公司里。Xerox公司于1970年建立了Palo Alto研究中心(PARC)，其中一项就是资助开发产品，以使得公司能进入计算机界。也许PARC最著名的设想家是Alan Kay（生于1940年），当他14岁的时候碰巧看到了Robert Heidein的小故事里描述的Van Bush的微缩胶片图书馆，并已设想了一种他称为Dynabook的轻便计算机。

PARC的第一个大工程是Alto，于1972年～1973年期间设计和制造出来。按照那个年代的标准，这是一个给人深刻印象的产品。它是安装在地板上的系统单元，有16位处理器、2个3MB的磁盘驱动器、128KB内存（可以扩充到512KB），以及一个三个按钮的鼠标。在16位单片微处理器之前出现，Alto的处理器由大约200个集成电路组成。

Alto的视频显示器是它与众不同的地方之一。屏幕的大小和形状与一张纸差不多——8英寸宽10英寸高。它以光栅图形模式工作，有606个水平像素和808个垂直像素，这样总共489 648个像素。每个像素占1位存储器，即每个像素不是黑色就是白色。用于视频显示的存储器容量是64KB，占用部分处理器的地址空间。

通过写入到视频显示存储器，软件可以在屏幕上画出图画或显示不同字体和大小的文本。通过在桌上滚动鼠标，用户可以在屏幕上定位指针，与屏幕上的对象进行交互。视频显示器

并非像电传打字机那样按行显示用户输入，按行写出程序输出。视频显示器的屏幕是一个二维的、高密度的信息阵列和更直接的用户输入源。

在20世纪70年代末期，为Alto所写的程序显示出了许多令人感兴趣的特点。多个程序可放到窗口中并同时显示在屏幕上。 Alto的视频图像使得软件超出文本范畴，并真正反映用户的思想。图形对象（如：按钮、菜单及称作图标的小图画）成为用户接口的一部分。鼠标用来选择窗口或触发图形对象，执行程序功能。

这就是软件，不仅提供了用户接口而且已达到与用户亲密交互的程度。软件进一步扩展了计算机的应用领域而不再仅进行简单的数字操作。软件设计出来就是——引用Douglas Engelbart在1963年写的论文的标题——《for the Augmentation of Man's Intellect》。

在Alto上开发的PARC只是图形用户界面即GUI（graphic user interface）的开始。但Xerox公司并没有销售 Alto（如果要销售的话，价格将在 3万美元以上）。经过10多年，高售价的思想体现在一种成功的消费品上。

1979年,Steve Jobs和来自苹果计算机公司的小组参观了 PARC，看到的东西给他们留下了深刻的印象。但是，他们花费了三年多的时间才推出了具有图形接口的计算机，这就是 1983年1月推出的不太受欢迎的Apple Lisa。而在1年以后，苹果公司就推出了很成功的 Macintosh。

初始的Macintosh配备有Matorola 6800微处理器、 64KB的ROM、128KB的RAM、一个3.5英寸的磁盘驱动器（可以存储400KB）、一个键盘、一个鼠标和一个视频显示器。该视频显示器水平512个像素，垂直342个像素（CRT测量对角只有9英寸），总共175 104个像素。每一个像素用1位内存，颜色为黑、白，这样视频显示 RAM大约需要22KB。

最初的Macintosh硬件做得很好，但是很难进行变革。 1984年，出现了使Mac与其他计算机完全不同的Macintosh操作系统，那时通常称作系统软件，后来称为 Mac OS.

像CP/M或MS-DOS这样的基于字符的单用户操作系统不是很大并且没有扩展的应用程序接口（API）。正如第22章解释的那样，这些基于字符的操作系统所需要的是使用文件系统的应用程序。而像Mac OS这样的图形操作系统则非常大且有几百个 API函数，每一个都由一个描述该函数功能的名称来标识。

像MS-DOS这样的基于字符的操作系统只需提供几个简单的 API函数就能使得应用程序在屏幕上以电传打字机方式显示字符，但 Mac OS这样的图形操作系统必须提供一种方法才能使得程序可以在屏幕上显示图形。从理论上来讲，可以通过实现一个 API函数来完成，该函数使得应用程序可以设置某个水平和垂直坐标的像素的颜色。但是，实际证明这种方法效率不高且图形显示的速度很慢。

因而由操作系统提供完整的图形编程系统就显得非常有意义，这意味着操作系统需包含有画线、画矩形、画椭圆（包括圆）和字符的 API函数。线可以由实线或虚线或点组成；矩形和椭圆可以用不同的填充模式来填充；字符可以显示成不同字体和大小并具有不同效果，如：黑体和下划线等。图形系统负责确定如何在显示器上把这些图形对象作为点阵来显示。

图形操作系统下运行的程序使用相同的 API在计算机视频显示器和打印机上画出图形。因此，字处理应用程序在屏幕上显示的文档可以与打印出来的文档非常相似。这种特点称为WYSIWYG，是“ What you see is what you get (所见即所得) ”的英文缩写。这是喜剧演员Flip Wilson在Geraldine角色中所贡献的计算机行话。

图形用户界面的吸引力部分体现在不同应用程序的工作大致相同，且与用户经验关系不

大。这意味着操作系统还必须支持API函数从而使得应用程序可以实现用户界面的不同部分，如按钮和菜单等。GUI通常看起来是友好的用户环境，它对编程人员来说同样是很重要的环境。编程人员在原来的基础上就可以实现现代用户界面。

在Macintosh推出之前，几家公司就已开始创建用于IBM PC及其兼容机的图形操作系统。从某种意义上来讲，Apple开发人员的工作要容易一些，因为他们硬件和软件一起设计。Macintosh系统软件只支持一种类型的磁盘驱动器，一种类型的视频显示器和两种打印机。但在PC上实现的图形操作系统需要支持许多不同的硬件。

另外，尽管IBM PC很早（1981年）就已推出了，但多数人已习惯于用他们喜好的MS-DOS应用程序且没有准备放弃它们。PC机的图形操作系统需要考虑的一个重要方面是，要使得MS-DOS应用程序的运行就像是专门为新的操作系统设计的应用程序一样。（Macintosh上就根本不能运行Apple II软件，因为它采用了不同的微处理器。）

1985年，Digital Research公司（CP/M的后续公司）推出了GEM（图形环境管理器）；Visicorp公司(销售Visilalc的公司)推出了VisiOn；Microsoft公司发布了Windows 1.0版，它很快被认为是“视窗战争”中最有可能的胜利者。然而，直到1990年3月发布Windows 3.0，Windows才开始吸引大量的用户。从那时起，它的普及率不断提高。到今天，已有大约90%的微机上使用的操作系统是Windows。Macintosh和Windows除了具有相同的外在表现外，它们的API是非常不同的。

理论上讲，除了图形显示外，图形操作系统并不比字符操作系统需要更多的硬件，甚至不需要硬盘驱动器。最初的Macintosh没有，Windows 1.0也不需要。Windows 1.0甚至不需要鼠标，尽管每个人都认为用鼠标操作更容易一些。

然而(这儿一点也不奇怪)，随着微处理器越来越快，内存和外存越来越大，图形用户界面也越来越流行。越来越多的特点增加到图形操作系统，至使它们越来越大。今天的图形操作系统通常需要200MB的硬盘空间和32MB以上的内存。

图形操作系统的应用程序几乎没有是用汇编语言编写的。早期Macintosh上应用程序的流行语言是Pascal。对于Windows应用程序来说，流行语言是C。但PARC再次使用了一种不同的方法。大约从1972年开始，PARC的研究人员就在开发一种称为Smalltalk的语言，体现了面向对象程序设计，即OOP的概念。

通常，高级程序设计语言的代码（通常以set、for、if这样的关键字开头的语句）和数据（用变量来表示的数）之间有区别。毫无疑问，这种区别源自冯·诺依曼计算机体系结构。在这种体系结构里，要么是机器码，要么是机器码用于操作的数据。

而在面向对象的程序设计中，对象是代码和数据的组合。在对象中，数据存储的实际方法只能通过与该对象相关联的代码才能理解。对象通过发送或接收消息来与其他对象通信，它给一个对象发送指令或从那里获得信息。

面向对象语言通常有助于编写用于图形操作系统的应用程序，因为编程人员可以用与用户感知对象的同样的方式来处理屏幕上的对象（如：窗口和按钮等）。在面向对象语言中，按钮是对象的一个例子。屏幕上的按钮有一定的尺寸和位置，并显示一些文本或小的图画，所有这些都是与对象相关的数据。与对象关联的代码确定用户何时用键盘或鼠标按下按钮，并且发送一个标明该按钮被触发的消息。

然而，最流行的微机上的面向对象语言是传统的类ALGOL语言的扩展，如C和Pascal。最

流行的由C扩展的面向对象语言是C++（前面讲过，两个+是增量操作）。C++大部分是由贝尔实验室的Bjarne stroustrup（生于1950年）完成的，开始作为转换程序，用来把用C++编写的程序转换成C程序（尽管C程序很难看也很难读），C程序可以像通常一样编译。

当然，面向对象语言并不能比传统语言多做些什么。但是编程是解决问题的方式，而面向对象语言使得编程人员能够考虑那些在结构上通常更好的不同的解决方法。也可以——尽管不是那么容易——用面向对象语言编写程序，编译后可在Macintosh上或Windows下运行。这样的程序并不直接涉及到API而是使用称作API函数的对象。两个不同的对象定义用来编译用于Macintosh或Windows API的程序。

许多小型机上的编程人员不再用命令行编译程序。取而代之的是编程人员开始采用集成开发环境（IDE），即在一个方便的程序里集成有所需的所有工具并且该程序可像其他图形应用程序一样运行。编程人员还利用一种称作可视化编程的技术，通过鼠标汇集按钮及其他组件来设计交互窗口。

第22章中讲到了文本文件。这种文件只包含有ASCII字符，方便人们阅读。在使用基于字符的操作系统时，文本文件是在应用程序之间交换信息的理想工具。文本文件的一个最大优点就是它们是可检索的——即程序可以查看许多文本文件并确定它们中的哪一个包含有某一字符串。但是，一旦某个操作系统中有一个工具可用来显示不同字体、不同大小及不同效果（如斜体、黑体和下划线），则文本文件似乎就很不适用了。其实，许多字处理程序以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。

但是，可以同文本一起编码信息（如字体定义及段落编排），且仍然得到可阅读的文本文件。关键是选用一个转换字符来表示这些信息。在Microsoft设计的RTF（rich text format）中，作为在应用程序之间交换格式化文本的一种方法，花括号{}及反斜杠\用来封装信息，标明文本采用何种格式。

PostScript是把这种概念发挥到极致的一种文本格式。PostScript由Adobe系统的创始人之一John Warnock（生于1940年）设计。这是一种通用的图形编程语言，主要用来在高端计算机的打印机上画出字符或图形。

把图形结合到个人计算环境是越来越好、越来越便宜的硬件的直接结果。微处理器越来越快，存储器越来越便宜，视频显示器及打印机分辨率不断增加且具有更多种颜色，所有这些促进了计算机图形的使用。

计算机图形产生于两种不同方式，与早些时候为区分图形视频显示器所用的词一样：矢量和光栅。

矢量图形用直线、曲线及填充的域来生成图形，这是计算机辅助设计（或CAD）程序的领域。矢量图形在工程和结构设计中具有重要用途。矢量图形可以按元文件的格式存放到文件中。元文件是矢量图形制作命令的聚合，这些命令通常以二进制形式编码。

矢量图形采用直线、曲线及填充的域，因而非常适合于桥梁设计等，但不能指望它来实际显示建造的桥梁的效果。桥梁效果图是现实世界的图像，用矢量图形来表示太复杂，因而很难表示出来。

光栅（也称作位图）可用来解决这一问题。位图把图像编码成位的矩形阵列，该阵列对应于输出设备上的像素。就像视频显示器一样，位图具有空间度（或分辨率），即指图像按像素表示的宽度和高度。位图也有颜色度（或颜色分辨率/颜色深度），是指每一个像素对应的

位数。位图中的每一个像素用相同的位数来表示。

尽管位图图像是二维的，但位图本身只是一个字节流——通常从最上面一行像素开始，接着是第2行、第3行等等。

一些位图图像是使用为图形操作系统设计的画笔程序通过“手工”生成的，还有一些位图图像由计算机代码来生成。现在，位图经常用来表现现实世界的图像（如：照片），有几种不同的硬件可用来把现实世界的图像输入到计算机中，这些设备通常用到称作电荷耦合（CCD）的器件，它是一种半导体器件，在光照下会放电。一个CCD单元用来采样像素。

扫描仪是这些设备中最古老的一种。就像影印机一样，它用一行CCD扫过印制图像（如：照片）的表面。随着光的强度不同，CCD具有不同的电荷积累。与扫描仪一起工作的软件把图像转换成位图存放在文件里。

视频摄像机用二维CCD单元阵列来捕捉图像。通常这些图像录制在录像磁带上。但视频输出也可以直接送到视频帧输入器，这是用来把模拟信号转换成像素值阵列的一块板。帧输入器可以使用任何普通的视频源，如VCR或激光影碟机，甚至直接来自于有限电视盒。

最近，数字照相机价格已适合家庭购买，它看起来很像一般的照相机。但是，数字相机不用胶片，而是用CCD阵列来捕捉图像并直接存储到照相机的存储器中，然后传输到计算机中。

图形操作系统常常支持某种格式的位图文件的存储。Macintosh使用Paint格式，这个命名参考了创立这种格式的MacPaint程序。（Macintosh的PICT格式综合了位图和矢量图形，是它的首选格式。）Windows里的格式是BMP，它是位图文件的扩展名。

位图可能很大，采用一些方法压缩它们是很有用处的。这些工作都归入到计算机科学中称为数据压缩的范畴。

假设正在处理图像，如前所述，每个像素占3位。若有一个图片有天空、一栋房子和一块草坪，这样的图片可能就会有大片的蓝色和绿色。也许，位图的最上面一行是72个蓝色像素，如果有一些方法能够在文件中编码这72个数字，以表示蓝色像素重复72次，则位图文件可能会变得更小。这样的压缩称为行程编码，即RLE（run-length encoding）。

一般办公用的传真机采用RLE压缩方法，在通过电话线传送之前压缩图像。由于传真机只把图像看成黑色和白色而没有灰度和彩色，因此，通常有很长串的白色像素。

已经流行了10多年的位图文件格式是图形交换格式即GIF，由Compu Serve公司于1987年开发出的。GIF文件采用称为LZW的压缩技术，“LZW”代表它的创建者：Lemplel、Ziv和Welch。LZW比RLE效果更好，因为它检测不同像素值的模式而不仅仅是针对具有相同值的像素的连续串。

RLE和LZW都是无损失的压缩技术，因为从压缩数据中可以重新生成完整的初始文件。换句话说，压缩是可逆的。很容易证明可逆的压缩方法并不适用于所有类型的文件。在某些情况下，“压缩”文件比初始文件还要大。

最近几年，有损失的压缩技术很流行。有损失的压缩是不可逆的，因为某些初始数据被丢弃了。不要用有损失的压缩技术来压缩电子报表或文字处理文档，因为每一个数字或文字也许都是很重要的。但这并不妨碍用有损失的压缩技术来压缩图像，因为只要丢弃的数据不会使得图片的整体效果有太大差别即可。这就是有损失的压缩技术用于可视心理研究的原因，它可以研究人的视觉，以确定什么重要，什么不重要。

最重要的用于位图的有损失的压缩技术统称为 JPEG。JPEG表示联合图像专家组（joint photography experts group），它描述了几种压缩技术，一些是无损失的，一些是有损失的。

把元文件转换成位图文件很简单。因为视频显示存储器与位图在概念上是一致的。如果一个程序知道如何在视频显示存储器中画一个元文件，则它也知道如何在位图上画元文件。

但是，把位图文件转换成元文件就没那么容易，有些复杂的图像甚至不能转换。与这项工作相关的一项技术是光学字符识别，即 OCR（optical character recognition）。如果一个位图上有一些字符（从复印机来的，或扫描页面得到的）并且需要转换成 ASCII码，就可用OCR。OCR软件需要分析位的模式并确定它们代表什么字符。由于这项工作的算法很复杂， OCR软件并不是100%准确。即使有些不准确，OCR软件也试图把手写体转换成 ASCII码字符。

位图和元文件都是用数字表示的可视信息。音频信息也可以转换成位和字节。

1983年，随着激光唱机的出现，数字化音响激起了消费者的热情，它也成为了最大的电子消费品。CD由Philips和Sony公司开发，在一个直径12cm的盘上一面可存储74分钟的数字化声音。之所以选择74分钟是因为贝多芬的第九交响曲刚好可以放在一张 CD上。

CD上的声音编码采用脉冲编码调制技术，即 PCM (pulse code modulation)。不管它的名字多么奇怪，从概念上讲，PCM是很简单的处理过程。

声音是振动产生的。人们的声音是振动，大号的声音是振动，森林里树倒下的声音也是振动，它们使得空气分子移动，空气一会儿挤压一会儿弹开，一会儿压缩一会儿放松，一会儿向后一会儿向前，每秒钟进行着成百上千次运动。空气最终震动耳膜，使得我们能够听到声音。

声波可以用1877年爱迪生的第一台电唱机上用来录制和播放背景音乐的锡箔圆桶表面上的凸起和凹陷来模拟。直到出现 CD之前，这种录制技术也很少改变，虽然圆桶换成了盘片，锡箔换成了塑性材料即塑料。早期的电唱机是全机械的，后来使用电子放大器来放大声音。麦克风上的可变电阻把声音转换成电流，喇叭中的电磁铁把电流转换回声音。

代表声音的电流并非本书中所讲的 1/0数字信号。声波是连续变化的，而产生这种电流的电压也是如此。电流是声波的模拟。一种称作模拟数字转换器（ADC）的器件 —— 通常在一个芯片上实现 —— 把模拟电压转换成二进制数。 ADC的输出是若干位数字信号 —— 通常为8、12或16位 —— 用来表明电压的相对级别。例如，12位ADC把电压转换成000h～FFFh之间的数，从而区分4096个不同的电压级别。

在脉冲编码调制这种技术里，代表声波的电压按照恒定的速率转换成数值。这些数以小孔的形式刻在光盘表面，从而存储在 CD上。通过从CD 表面反射的激光可以读出这些信息。在播放的时候，这些数又通过数字/模拟转换器即DAC转换成电流。（DAC也用在彩色图形板上，用来把像素值转换成模拟的彩色信号送到显示器。）

声波电压以恒定的速率转换成数字，该速率称为采样速率。1928年，贝尔实验室的Harry Nyquist证明了采样速率至少为需要记录和播放的信号的最大频率的两倍。通常认为人们能听到的声音的频率范围为 20～20 000赫兹。CD所用的采样频率比最大频率的两倍还要大一些，定义为每秒采样44 100次。

每个样本所用的位数取决于CD的动态范围，即记录和播放的声音的最高频率与最低频率之差。这有些复杂：电流不断地变化来模拟声波，尖峰称为声波的振幅。我们所感受到的声音强度是振幅的两倍。 1贝尔（bel）表示强度的10倍增强；1分贝(decibel)是1贝尔的1/10，表示人们所能感受的声音的几乎最小的强度变化。

每个样本用16位表示，这样可以表示96分贝的动态范围，差不多是从能听到的声音的阀值（低于这一值则不能听见）到能忍受却不感到痛苦的声音的阀值的差。 CD盘中用16位表示一个样本。

所以，CD盘中每秒声音有44 100个采样样本，每个样本2个字节。立体声则需要两倍的采样信息即每秒总共176 400字节，每分10 584 000字节。（现在可以知道为什么在20世纪80年代之前声音的数字记录不是很普遍。）CD上74分钟的立体声需要783 216 000字节。

数字化声音与模拟声音相比具有很多众所周知的优点。特别是，无论何时复制模拟声音（例如从录音磁带生成电唱片）都会有一些失真。而数字化声音是数字信息，总可以如实地转录和复制。过去常常是电话信号传输线路越长则声音越糟。现不再是这样了，因为现在许多电话系统都是数字的，跨越一个国家的呼叫信号就像跨越一条街道一样清晰。

CD也可像存储声音一样来存储数据。用得最广泛的用来存放数据的 CD称作CD-ROM（CD只读存储器），通常CD-ROM最多可存储约660MB。今天，许多计算机中都装有CD 驱动器，许多应用程序和游戏都在CD-ROM中。

大约10年前，声音、音乐、视频开始进入个人计算机中，这称为多媒体。现在多媒体已经很普遍了，也不需要特别的名称。今天出售的许多家用计算机有声卡，内含一个 ADC用来把从麦克风来的声音录制成数字，还有一个 DAC用来通过喇叭播放录制的声音。声音可以以波形文件存放在磁盘中。

因为在家用计算机中录制和播放声音并不总是需要达到 CD的质量，所以 Macintosh和Windows提供低的采样速率，如22 050、11 025和8000赫兹，以及较小的8位样本信息和单频度录制。声音以每秒8000字节来录制，即每分480 000字节。

人们从科幻电影和电视中知道，未来的计算机可以用英语与用户交谈。一旦计算机有了数字化录制和播放声音的硬件，则所有通向这一目标的其他工作就可用软件来完成。

使计算机能讲人们能识别的单词和句子的方法有两种。一种方法是让人们录制句子段落、短语、单词及数字，然后存储在文件中，并且用不同的方法串在一起。这种方法通常用在通过电话访问的信息系统中，它在只需播放有限的单词和数字组合的情况下能很好地工作。

一种常见的声音合成形式涉及到一个用来把 ASCII码字符转换成波形数据的进程。例如，由于英语拼写并不总是一致的，所以这样的软件系统用一个词典或复杂算法来确定单词的确切发音。基本的音节（称作音素）组合成整个单词。通常软件需要做一些调整，例如，如果一个句子后面跟着问号，则最后一个单词的声音频率必须增加。

声音识别 —— 把波形数据转换成ASCII码字符 —— 是一个更复杂的问题。的确，许多人在理解口语的方言方面有一些问题。在个人计算中使用听写软件时，通常需要训练以便能合理转录某个人所说的话。其中涉及的一个问题已超出了转换成 ASCII码文本的范围，即编程使计算机“理解”所说的话。这个问题是人工智能的研究领域。

今天，计算机中的声卡也提供小的电子音乐合成器，它能模仿 128种不同的音乐乐器和47种不同的打击乐器，称作MIDI合成器。 MIDI即乐器数字接口，在20世纪80年代早期由电子音乐合成器制造者协会开发出来，用来把这些电子乐器互相连接起来并连到计算机中。

不同种类的MIDI合成器用不同的方法来合成乐器的声音，其中一些比另一些更逼真。MIDI合成器的性质已远远超过了MIDI定义的范畴。所要做的无非是通过演奏声音来响应短消息 —— 通常长度为1、2或3字节。MIDI消息常常指明需要什么乐器、将要演奏哪个音符，或正

在演奏的音乐要停止演奏。

MIDI文件是加上时间信息的MIDI消息的集合。通常，一个MIDI文件包含有计算机上的MIDI合成器所能演奏的所有音乐成分。要包含同样的音乐，MIDI文件通常比波形文件小得多。按照相对大小来说，如果说一个波形文件像位图文件，则 MIDI文件就像矢量图形元文件。MIDI文件的不足之处在于：以MIDI文件编码的音乐可能在一个MIDI合成器上演奏得很好，但在另一个合成器上演奏出来却很糟。

多媒体的另一个特征是数字化电影。电影和电视图像的移动效果可以通过快速显示一系列静止图像来达到。这些单个图像称为帧。电影以每秒 24帧的速率来播放，北美电视每秒为30帧，世界上其他许多地方的电视每秒为25帧。

计算机中的电影文件由一系列有声音的位图简单组成。但如果不经过压缩，一个电影文件将包含大量的数据。例如，假设一个电影每一帧的大小是 640×480像素的计算机屏幕，有24位彩色，则每帧有921 600字节。按每秒30帧，则每秒27 648 000字节。一直乘下去，则每分钟为1 658 880 000字节，一个两小时的电影有199 065 600 000字节，大约200GB。这就是为什么许多在个人计算机上播放的电影又小又短又跳跃的原因。

JEPG压缩方法用来减少存放静止图像所需的数据量，而 MPEG压缩方法用于存放运动图像。MPEG代表移动图像专家小组。移动图像压缩技术利用的是这一事实，即某一帧通常包含从前一帧复制来的大量信息。

对不同的媒体来说，有不同的MPEG标准。MPEG-2用于高清晰度电视（HDTV）及数字视盘（DVD），也叫数字万用盘。DVD的大小与CD一样，但可以两面记录且每一面有两层。在DVD中，视频信息按照大约50倍这样的因子进行压缩，所以，一个两小时的电影只需 4GB，且只需放在一面的一层。如果用两面和两层，则 DVD的容量可达到大约16GB，约是CD容量的25倍。可以预见，DVD最终将取代CD-ROM来存储软件。

CD-ROM和DVD-ROM是不是Vannevar Bush的预言在今天的实现？他开始设想的 Memex是用缩微胶片，但用CD-ROM和DVD-ROM更适合。电子媒体比物理媒体具有优越性，因为前者更容易检索。遗憾的是，很少有人同时访问多个 CD或DVD驱动器。我们所接触的Bush概念中的文件柜并不涉及存储桌面上所需的所有信息，它涉及的是互连计算机使得它们共享信息并更有效地利用存储空间。

公开从远程操作计算机的第一人是George Stibitz ，正是他在1930年设计了贝尔实验室的继电器计算机。继电器计算机的远程操作于 1940年在 Dartmouth进行了演示。

电话系统是用来在线路上传输声音的，而不是位。电话线路上传输位需要将位转换成声音然后再转换回位。一种频率和一种振幅的连续声波（称作载波）并不能传送真实的信息。但是，如果改变声波的一些东西 —— 换句话说，在两种不同的状态之间调制声波 —— 则可以表示0和1。在位和声波之间的转换由称作调制解调器的设备来实现。调制解调器是串行接口的一种形式，因为一个字节的位是一个接一个传输的，而不是同时传输的。（打印机通常通过并行接口与计算机连接：8根线同时传输一个字节。）

早先的调制解调器采用称作频移键控（FSK）的技术。以300bps 传输的调制解调器把0调制到1070赫兹，把1调制到1270赫兹。每个字节以一个起始位开始，以一个停止位结束，所以每个字节需要10位。以300bps的速率传输，每秒只传输30个字节。许多现代调制解调器用更高级的技术能达到超过100倍的速率。

早期家用计算机爱好者可以用计算机和调制解调器建立公告牌系统（ BBS），其他计算机可以接入并下载文件，即从远程计算机传输文件到自己的计算机。这种概念扩展到了如CompuServe这样的大型信息服务。在大多数情形中，通信完全采用 ASCII码字符形式。

Internet则不同于这些早期的成就，因为它是分散的系统。 Internet其实就是计算机之间相互通信的协议集合，其中最主要也是最重要的是 TCP/IP，由传输控制协议（ TCP）和网际协议（IP）组成。与通过线路只传输 ASCII码字符不同，TCP/IP的发送程序把大的数据块分割成小的包，在传输线路（通常是电话线）上独立传输，在另一端重新装配。

Internet上流行的图形部分是World Wide Web，采用HTTP，即超文本传输协议。在Web页面上看到的数据由称作 HTML即超文本标记语言的格式来定义。这些名词中超文本这个词用来描述相关信息的链接，非常类似于 Vannevar Bush提到的Memex。一个HTML文件可以包含到其他Web页面的链接，从而容易地访问它们。

HTML与前面讲到的富文本格式（RTF）很相似，都包含有带有格式信息的 ASCII码文本。HTML也可包含GIF文件、PNG（portable network graphics）文件和JFIF（JPEG文件交换格式）文件等格式的图形。许多 World Wide Web浏览器可以浏览HTML文件，这是文本格式的一个优点。把 HTML文件定义成文本文件的另一个优点是它更容易查找。不管它的名称如何，HTML并不是像我们在第 19章和第24章讲到的那些真正的程序设计语言。 Web浏览器读取HTML文件并依照它来编排文本和图形格式。

当你在浏览某个 Web页面并在上面操作时执行一些特殊的程序代码是有用的，这些代码可以在服务器（指那些存储初始 Web页面的计算机）或客户机上运行，客户机即自己的计算机。在服务器端，通常所要做的全部工作（例如对客户端填写的在线表格的解释）可以通过公共网关接口（CGI）脚本来处理。在客户端，HTML文件可以包含简单的程序设计语言，如Java Script。Web浏览器就像解释HTML文本一样来解释Java Script语句。

为什么一个Web站点不能简单地提供一个可以在你的计算机上执行的程序呢？这涉及到一个问题，你的计算机是什么？如果是 Macintosh，则需要一个包含 PowerPC机器码的可执行文件并使用Mac OS API; PC兼容机需要一个包含 Intel Pentium机器码的可执行文件，并使用Windows API。但还有其他计算机及图形操作系统。而且，你也不想不加选择地下载可执行文件，它们可能来自于不值得信赖的地方且带有某种恶意。

对这些问题的回答可由 Sun公司的 Java语言来提供（不要与 JavaScript混淆）。Java是一个完美的面向对象的程序设计语言，非常像 C++。前面几章里已经解释了编译语言（产生包含机器码的可执行文件）和解释语言（不产生可执行文件）之间的区别， Java介于两者之间。Java程序要经过编译，但编译的结果不是机器码，而是 Java字节码。在结构上Java字节码与机器码很相似，但用在虚构的计算机即 Java虚拟机（JVM）上。执行编译后的Java程序的计算机模拟JVM解释Java字节码。Java程序可在不同机器上的不同图形操作系统上运行，所以是具有平台独立性的程序。

虽然本书着重讲了用电信号在线路上传输信号和信息，但一种更有效的方式是通过光纤 —— 由玻璃或聚合体制造的小管道，可从不同角度传输光信号 —— 来传输光信号。通过光纤传输光信号可以达到以吉赫计算的数据传输速率 —— 即每秒几百万位。

所以，似乎是光子而不是电将要负责未来家庭和办公室的大量信息传输，它将比摩尔斯电码的点划更快，也比那些我们曾用来午夜与好朋友通信而精心设计的闪灯更快。