HOPE

用Turbo C开发三维图形软件

李春葆 译
马 宁 校

北京希望电脑公司

用 Turbo C
开发三维图形软件

李春葆　编译
马　宁　审校

- 全部软件均用Turbo C开发
- 给出了单个物体部件的开发方法
- 给出了集成图形软件系统D3D的研制过程
- 列出所有软件的源代码
- 说明了D3D的使用方法

说　明

本书作者是美国著名的计算机图形学专家，它著有另外五本图形学专著。

本书主要讨论一个三维图形集成化软件D3D的研制过程。该软件具有三维图形的旋转、平移和隐线消除等功能，并可把多个物体部件组合起来构成复杂的物体。该软件使用灵活方便，适用于机械零件CAD等领域。同时作者讨论了单个物体部件如球体、柱体、链体、绳和结等的设计方法，给出了B-样条曲线、曲面逼近的实用程序，以及支持HP(惠普)绘图仪的程序PLOTHP等。

全书分为三部分，第一部分介绍了D3D软件的使用方法，即用户手册，第二部分讨论了单个物体部件的设计方法，第三部分讨论D3D软件的研制过程，两个附录列出了D3D软件的全部源代码。

目　录

第一部分　用户手册

第二部分　编制应用程序

第三部分　图形程序设计

第一部分　用户手册

第一章　设计三维图形

1.1　三维坐标系统

本书主要讨论用于设计三维物体图形的程序D3D。该程序源码用Turbo C编制，全部程序清单附在本书末尾。你可以不必把这些源程序敲入计算机内，而直接从本书出版商那里购置源代码和可执行代码。这样，即使你不熟悉C程序设计语言或没有C编译器，照样能够使用该程序。如果要了解该程序是如何运行的，可以参阅本书的第三、四两章，不过需要了解C程序设计语言和所使用的机器。如果不具备程序设计经验，就必须阅读第一、二两章，它们是专门为D3D用户编写的。阅读时需有一些最基本的数学知识。

在科学和工程中使用的三维坐标系统通常是如图1.1所示的直角坐标系统，它由x、y和z三轴组成，其两两相互垂直。三个坐标轴均穿过源点o，其长度没有限制（无限长）。图1.1中画出的只是三个坐标轴的很小一部分（均为正方向），我们称之正坐标轴。坐标系统中的坐标均为实数，点p的直角坐标为x_p、y_p、z_p，其意义是：从源点o开始沿x轴方向走x_p距离，然后沿y轴方向走y_p距离，最后沿z轴方向走z_p距离便到达p点。

我们称如图1.1所示的系统为右手系，因为把x轴以z轴为轴线沿右手系方向旋转90°便与y轴重合，采用这种方向用右手旋动螺丝时，螺丝会慢慢向z轴正方向移动。把一个右手系三维直角坐标放在空间中存在各种形式，在图1.1中，我们采用z轴向上，x和y轴在水平面上，称之为xy平面。

除直角坐标系统外，球坐标系统也是很有用的。它同样采用三个实数来指定点p的位置，只是不再用x_p、y_p和z_p，而是采用希腊字母ρ θ和φ，如图1.2所示，ρ是点p和源点o之间的距离，换句话说，ρ是以o为球心穿过p点的球的球半径。

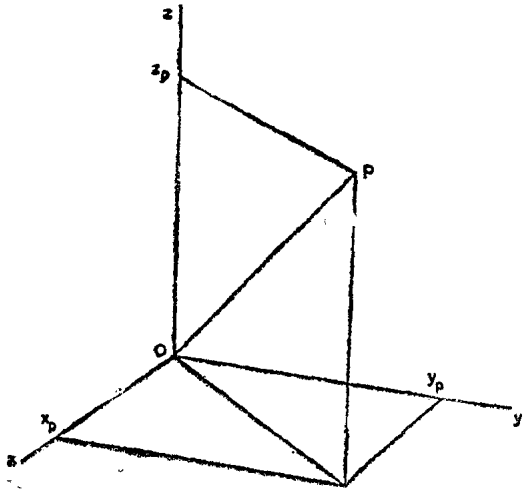


图1.1　直角坐标

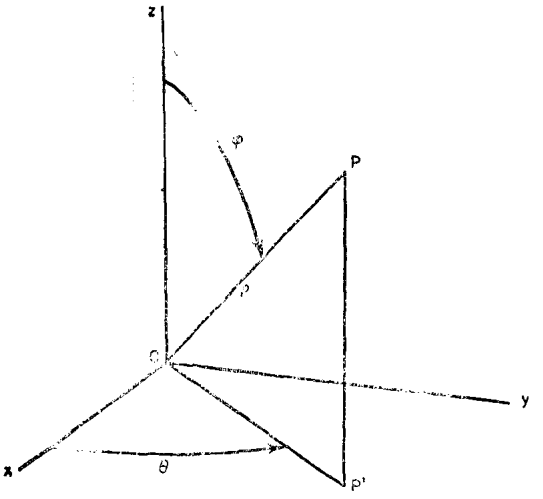


图1.2　球坐标

符号θ和φ表示角度，如图1.2所示，为了确定xy平面上的θ，使用p在该平面上的投影p′点，p和p′之间的连线垂直于xy平面。θ便是把x轴逆时针方向旋转到与p′点重合的所旋转过的角度，例如，若p′落在第一象限内，则θ在0°到90°之间。

φ表示z轴和op线段之间的夹角，φ的范围在0°和180°之间。如果你熟悉三角函数，很快发现这两个系统坐标存在如下关系：

$$x=\rho\sin\varphi\cos\theta$$
$$y=\rho\sin\varphi\sin\theta$$
$$z=\rho\cos\varphi$$

1.2 物体、视点和透视图

使用 D3D 处理点、线段和其它图形，一个实体总是以平面为界的。曲面可以由一系平面来逼近，类似于曲线可由一系列直线逼近一样。因此，一个边界面可以是任何多边形，也可能有孔。如果要计算机产生透视图，就要以某种方式给出这些多边形面的顶点。为此，我们使用1.1节所讨论为右手系直角坐标系统，你不必当心计算机屏幕能否显示出整个物体的视图，所有比例变换（Scaling）和位置都是自动实现的。你可以使用任何长度单位，例如英寸，米或公里等，但是一旦选择了某个单位，所有点坐标都必须使用这种单位。我们眼睛所在的位置称为视点，用字母E表示视点。E与物体的中心点o之间的连线Eo称为视中线，从E到o的方向称为视力方向，如图1.3所示，我们在一个圆锥体中能够看到整个物体。

为了说明视点E和物体之间关系，我们想象一个新的坐标系统：物体中心点 o 作为源点，每个轴与原来对应的轴平行，我们给出视点E的球坐标ρ、θ和φ与新坐标系统之间的关系，ρ是如图1.3所示视中线段EO的长度，换句话说，它是视距离。如图1.3给出了投影面，它是一个没有厚度的透明板，是假设的放在观察者和他所看的物体上点或一系列点之间的平面，投影面垂直于观察者的视中线，从眼睛视点E到物体上点的连线称为视点，所有视线与投影面的交点构成了物体的透视图。把一个物体投影到一个平面称为中心投影，因为所有投影线都通过视点E。

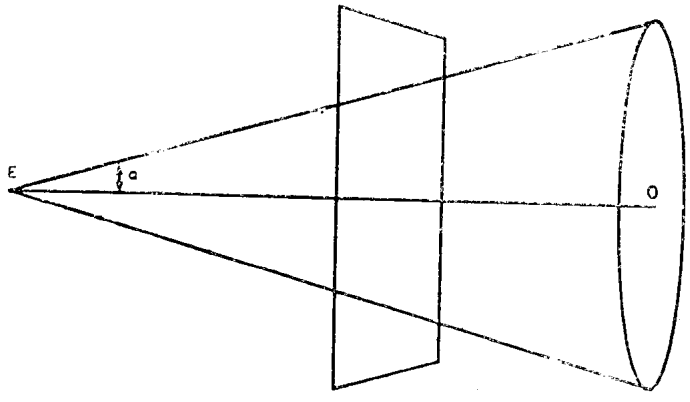


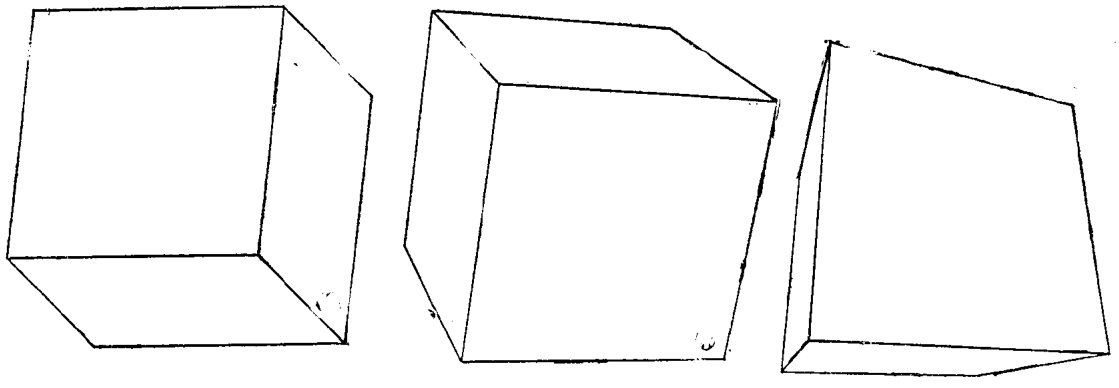
图1.3 圆锥体和视线

很清楚投影面和视点E之间的距离确定透视图的大小。程序D3D自动选择这个距离以便整个物体的透射图能够放在屏幕上，因此我们不必担心投影面的位置。

圆锥体的中心轴与其边界线之间夹角α一般较小以便该视图满足人们的需要。例如，如

果要显示边长为 1 的立方体，通常取ρ≥5，如果取值太大如ρ=100000，相应的视图并不比较小ρ值对应为视图小，这是为什么总是不把视线距离取得太大的原因。如果ρ为无穷大，即物体位于无穷远处，我们得不到真正的视图，但在物体上所有的平行线在视图上边是平行的。这可以通过如下想象得到：假设图1.3中的α很小，以至于从物体各个点到达视点E的视线都是平行的。因此可以用很大的ρ逼近于平行投影，这用于实际绘画中，因为它比透视法更容易采用。如图1.4所示，对应于不同的ρ值，我们看到边长为 1 的立方体有三种表示方式。

很多人更习惯于图1.4(a)，它采用ρ=5，如果把它放在很远处会产生线段按透视法缩短的透视效果。图1.4(b)通过取ρ=100000得到，其透视效果不明显。平行立方体边右图1.4(b)中边是平行的，把它放在很远处，线段不会按透视法缩短。严格地说，仍存在透视效果的，因为ρ虽大，但毕竟不是无穷大，只是这种透视效果可以忽略不计。这里我们把平行投影作为透视投影的一个特殊情况，而非另外的问题。因此我们的程序D3D不仅适用于透视图，边适用于如图1.4(b)的平行投影。图1.4(c)的立方体取ρ=2．它看起来不那么自然，



(a)　　(b)　　(c)

图14．(a)ρ= 5　(b)ρ=100000　(c)ρ= 2

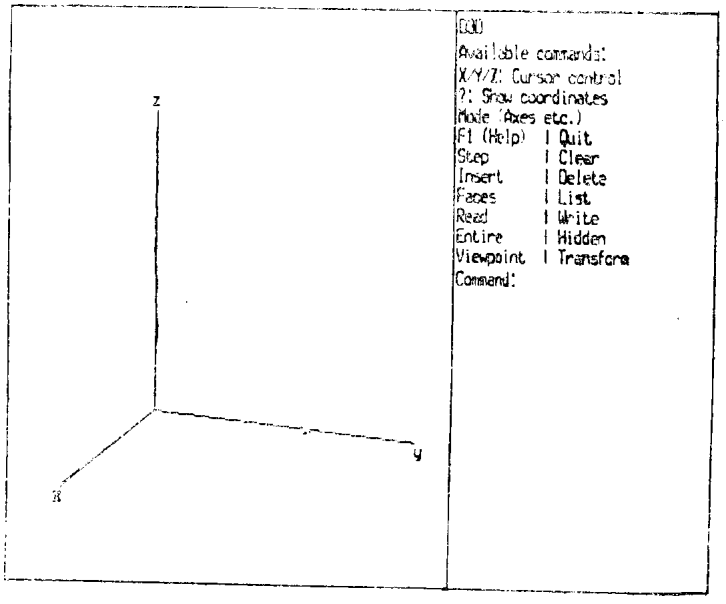


图1.5　初始屏幕

由于ρ太小，过分地夸大了透视效果。因此也要小心不要把ρ取得太小。对于θ和φ，我们可以选取任何值，如果保持ρ和φ不变，θ由0°增大到360°，也就是我们的眼睛在水平面上绕物体转一圈。如果选取φ=0°，视点便在物体的正上方，所得视图为俯视图，如果φ=90°，那么视点是水平方向上，大多数情况下，φ取45°到90°之间，也就是我们的眼睛在比物体高的某个地方看物体，如图1.4都属于这种情况。本程序中我使用了θ=0°，φ=70°。

如果你有了系统磁盘，可以直接使用文件EXAMPLE1.DAT做各种视点类型的实验。我们键入如下命令启动该程序：

```
D3D
```

或者

```
d3d
```

本书之后我们通常使用命令（D3D程序提供的各种操作命令）的第一个大写字母，它很容易与文本中同行的其它小写字母区分开来。如果愿意，你也可以使用相对应的小写字母。

启动D3D程序后，屏幕出现一些信息，读完这些信息后，按Enter键，便出现如图1.5所示的屏幕。我们看到，该屏幕分成两部分，一个用于显示三维图形，另一个显示命令菜单、信息提示和从键盘上输入的数据，后者称之为信息区域。

为了快速显示图形，我们现在直接从已存在的文件中读取数据，现在按：

```
R
```

它是读命令，之后出现如下信息：

```
Input file
```

这提示我们输入一个已包含输入数据的文件名，假设现在输入文件名：

```
EXAMPLE1.DAT
```

（输完后按Enter键），我们看到如图1.6(a)所示的屏幕。

你想知道为什么所有线不一样粗，这里用透视效应，越接近于视点的线显示得也越粗，这可以帮助我们更好地观察物体的线型模型，我承认在有些情况下这种效果并不一定很好，你可以在最终结果中以同样粗细显示线型模型的所有线段。我们在 1.5节看到的就是这种视图。

产生图1.6(a)后，我们可以通过改变视点得到同样物体的其它视图，该点的位置用如图1.2所示的球坐标给定。程序开始时，视点（缺省值）为ρ=1000，θ=20° φ=75°。我们可以把视点放在物体内部空间的任何地方，因而产生各种视图，使用如下命令改变视点：

```
V
```

屏幕立即变成类似于图1.2所示的样子，我们要记住视点的球坐标ρ、θ、φ的意义，然后系统显示当前的值，我们可以改变它们，如果直接按Enter键，则保持原来值不变。注意，其中θ和φ表示成度数，一旦以这种方式处理ρ、θ和φ后，便显示对应于新视点的视图。这样，如果选取很大的ρ如ρ=100000，我们会得到平行投影的效果，除此之外，我们还可产生物体的正面视图、侧面视图和俯视图，如图1.6(b)(c)和(d)所示，这在工程绘图中是很有意义的，这里选取很大的ρ值，θ和φ值分别如下：

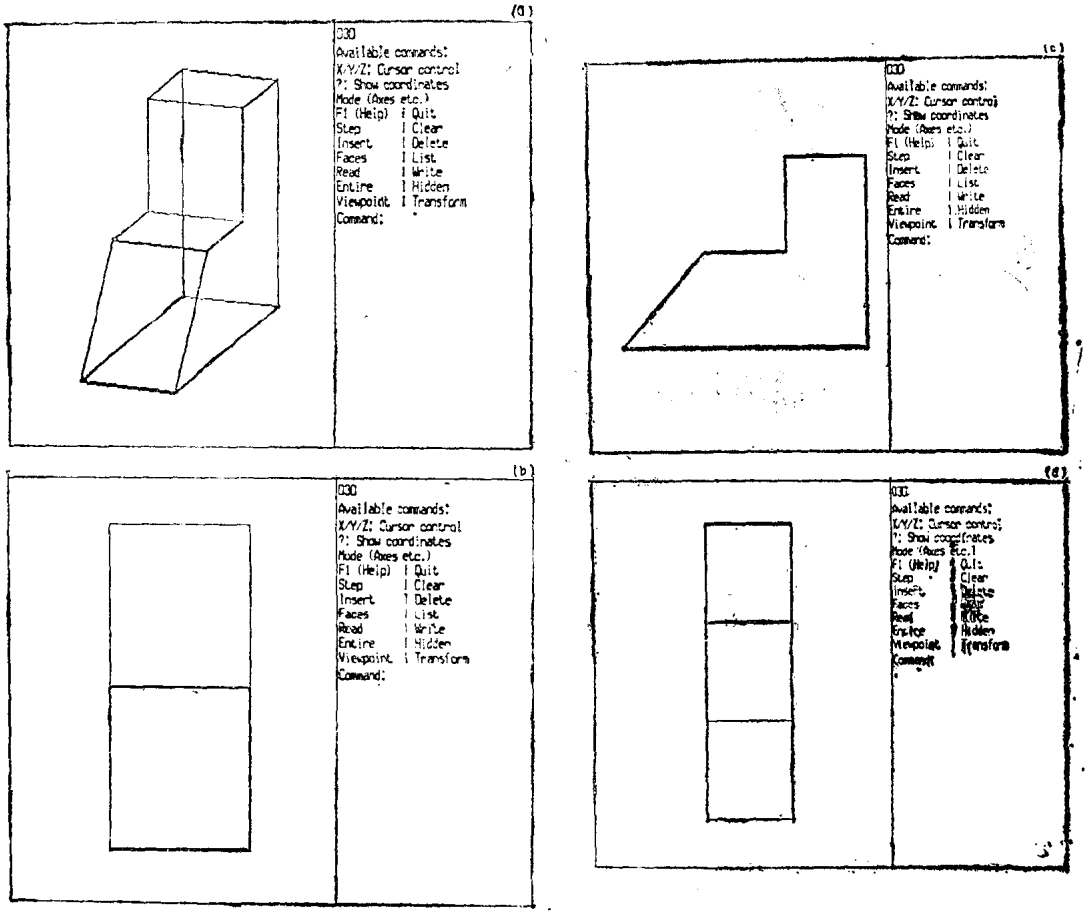


图1.6 (a)透射 (b)正面图 (c)侧面图 (d)俯视图

正面视图（从x轴正方向看到的视图）：$\theta=0°$，$\varphi=90°$

侧面视图（从y轴正方向看到的视图）：$\theta=90°$，$\varphi=90°$

俯视图（从z轴正方向看到俯视图）：$\theta=0°$，$\varphi=0°$

这些视图在工程中很有用，不仅因为它们容易手工绘制，而且视图中所有与视中线垂直的线段保持长度不变（不受透视的影响）。

本节中我们学会使用程序D3D来启动它，我们也要知道如何退出该系统，如很多其它程序一样，我们使用命令Quit，简写为：

Q

达到这个目的。

1.3 点操作和光标控制

现在我们讨论如何设计和画出新的物体视图，而不是象前节那样直接从文件里读取一个已存在的物体数据。我们知道，任何三维物体都可以表示成直线段，而线段又是由其两个端点决定的，因此我们首先讨论如何处理点。一旦确定了某个点，就要把它插入到三维空间中去，这一过程是使用 I 命令完成的。实际上这个命令在屏幕上的提示信息为Insert，我们只

需按第一个字母：

```
I
```

系统便提示要输入 4 个数：

```
n x y z
```

如下是输入点的一个实例：

```
5 1.49 0.8 3
```

输入了后，按Enter键，计算机知道它是最后一个数字。以后我们不再解释这个插入状态的意义。记住，如果等了很长时间仍无任何反应，你可再按Enter键（如果多按了该键，系统立即出现信息：Invalid Command，你可以简单地忽略它）。数字x、y和z是对应直角坐标中一个新的点，数字n是一个正整数，作为点的编号，以便识别这个点。我们先输入该点编号，然后输入三个坐标值x，y和z，最后系统显示出这个新插入的点（点5），如图1.7所示。除了该点外，我们还看到一条从该点到xy平面的垂直线，系统为了使你清楚地看到点（x，y，z）在三维空间中的哪个地方，把它与xy平面上的附加点（x，y，o）连接起来，也就是如图中点5所示的那样。

在构造一个新的物体的过程中，系统显示每个点编号和三根坐标轴，但在最终形成的视图中并不需要它们，你可以用如下命令显示或消除它们：

```
M
```

该命令将在1.4节末尾讨论。

如果我们要插入多个点，并不需要每次都使用 I 命令，只要不按其它命令字母，D3D总是保持点插入状态。如果插入几个点后，已使用了其它命令，若要恢复插入状态，只需重新按 I 键即可。否则，在插入第一个数字时出现如下错误信息：

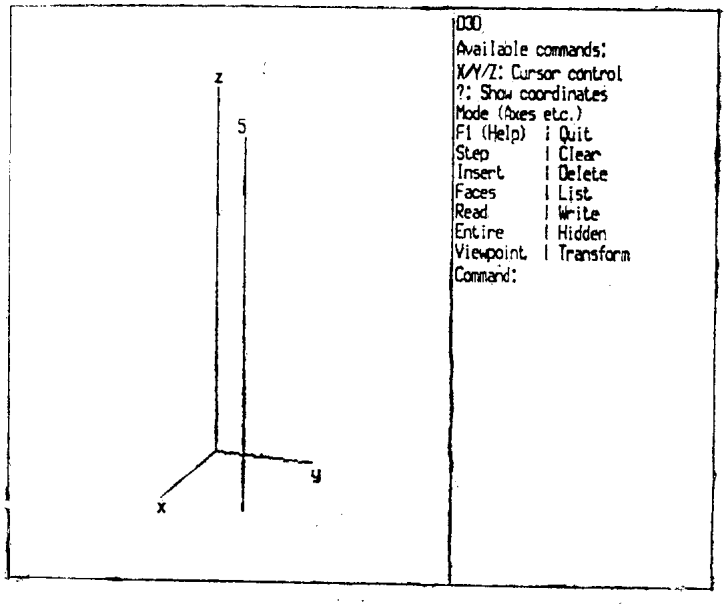


图1.7 用命令I插入点

```
Don't begin with digit
```

可以使用Delete命令删除描入的点，该命令可简写为：

```
D
```

按了这个字母后，系统将自动删除两个界限所确定的范围内的所有点。因此需要输入这两个界限，即一个下界限和一个上界限，所有比下界限大以及比上界限小的点编号对应的点都被删除。如果只要删除一个点，就在如下提示下输入要删除点的点编号：

```
Lower bound:
```

当出现以下提示信息（要求输入上界限），直接按Enter键：

```
Upper bound:
```

如果删除一个以上的点，系统将显示所产生的新的视图，该图中，已删除的点不复存在。

我们可能担心一旦误按了D键，会出现删除一个完整的物体视图的危险。其实D 键是很安全的。一旦误按D键后，有两种方法可以避免破坏原来的视点：一是在Lower bound提示下输入一个相当大（至少比视图中最大点编号大）的整数。另一种方法是连续按两次 Enter 键，这时系统便删除一个为空的点集。

如果已经插入了多个点，要想知道每个点的坐标，此时可直接打入问号：

```
?
```

系统出现提示：

```
point number?
```

当输入点编号后，系统便在信息区域里显示出相应点的坐标值。

我们还可以采用另外一种方式插入点。在很多情况下，后者比前面介绍的直接输入点坐标更方便。这里我们使用光标（cursor）控制，光标在屏幕上是以一个小正方形出现的。在二维图形中，用户使用四个光标键（上、下、左和右光标键）沿四个不同方向移动光标，而在三维图形中，存在六个不同方向，我们要沿每根轴的正、负方向移动光标。在D3D中，我们采用 x、y、z 键和＋、－键之间的相互组合来解决这个问题，在大多数键盘上都有两套加减与光标键，我们最好采用它们较接近的那一套（这避免shift键），如小键盘上的键。当我们按了x、y、z之一时，光标便停在屏幕坐标系统的源点处。要使光标沿相应轴方向移动，还要按＋或－键，如果要改变方向（不指从＋到－或从－到＋，而是指从x轴变成y轴等），则需键入相应轴的名字。

光标步长的缺省值为0.2，如果需要，我们可以通过按S键来改变它。例如，要定义一个坐标为（0.6，0.4，－0.2）的点，其光标控制的操作过程为：

```
X+++Y++Z-I
```

下面的表列出了光标移动过程中，连续光标点的位置。

按键	光标位置		
	x	y	z
X	0.0	0.0	0.0
+	0.2	0.0	0.0
+	0.4	0.0	0.0
+	0.6	0.0	0.0

```
Y       0.6     0.0     0.0
+       0.6     0.2     0.0
+       0.6     0.4     0.0
Z       0.6     0.4     0.0
-       0.6     0.4    -0.2
I       0.6     0.4    -0.2
```

其中的命令I是用来在当前光标所处位置插入一个新的点，系统自动给这个新点设置点编号，并把它显示在屏幕上，其编号是未使用的最小正整数。注意I键不改变当前光标所处的位置，例如，若下一个要移到的点位置为（0.6，0.4，−0.4），那么只需再按一次减号即可。

上面的例子看起来比较复杂，其实不然。首先，光标的当前位置都在屏幕信息区域里以数字坐标形式显示出来，因此我们可以根据这些信息控制光标移动。第二，在很多应用中开始时并没有给出点的坐标位置，要根据观测实际情况加入一些点，这样就不能采用前面介绍的I命令了。

有时需要把光标放在以前已插入点的位置上。最快的方式为：把光标移到最接近于要放到的点（相对于其它点），然后按：

```
J
```

该命令使光标立即跳到这个最接近的点。这解决了光标移动不很精确带来的麻烦。

现在我们看到，存在两个J命令，一个是主菜单中显示的Insert命令；另一个是在光标控制状态下使用。在后者状态下，也可以使用命令：

```
D
```

删除一个点，此时只能紧跟J命令后使用D，J命令保证光标准确落在要删除点的位置上，但只能一次删除一个。

如果要删除所有已插入的点，我们不必一个个删除，可以利用如下命令直接清除整个屏幕：

```
C
```

该命令是很有用的，例如我们要画新的物体视图，必须先用C命令删除所有已定义的点。当我们使用R命令从一个文件中读取一个物体数据之前，也需要使用该命令清屏。

1.4 线段、面和物体

三维物体可以采用两种方式显示，即所谓的线型模型和实物体模型，如图1.8所示，这里选取一个立方体，对于不同的视点，可以看到所有六个面（视点在物体内）或三个面（视点在物体外），或者所有十二条边或九条边。我们可以显示这两种图形，先显示线型模型，然后通过消除隐线段构造实物体图形。

让我们假设采用1.3节介绍的两种数据输入方式之一插入一个立方体的八个顶点，也就是说，要么明确地插入八个点的编号和相应的坐标，要么利用光标控制输入。前者在按J键后输入如下八个点：

```
1       1       1       1
2       0       1       1
3       0       0       1
```

4	1	0	1
5	1	1	0
6	0	1	0
7	0	0	0
8	1	0	0

采用光标控制方式可能更方便，这时，点在屏幕上是可见的，如图1.9所示（点在屏幕上的显示也和视点有关，在图1.9中，使用了视点的缺省值：ρ=1000，θ=20°，φ=75°）。

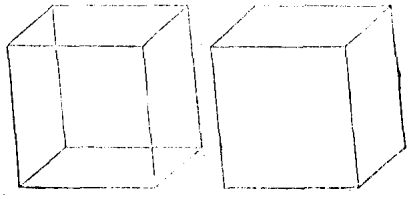


图1.8 线型模型和立体物体

我们还可以通过如下命令定义线段和面：

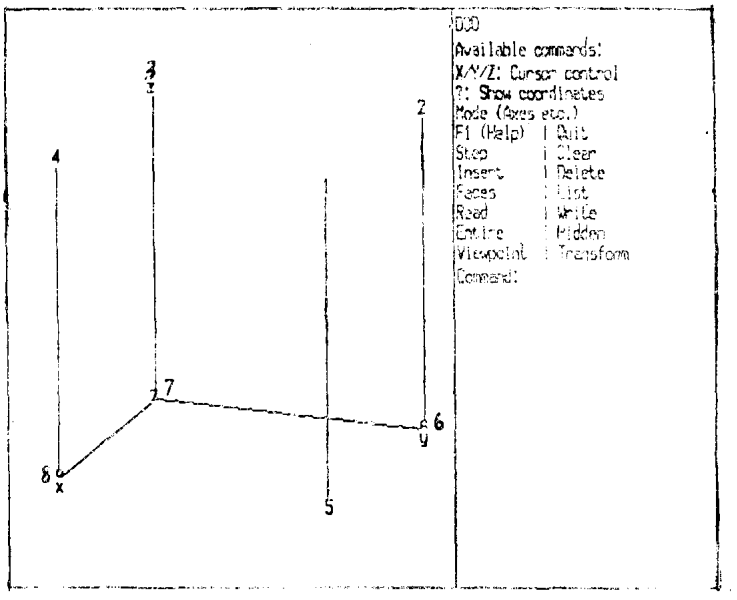


图1.9 立方体的顶点

F

使用该命令后，系统显示如下信息：

Nrs Closed by Period:

如果只要显示线型模型，则先给出立方体的两个水平面(每个面由4个顶点构成)，再给出4条由顶点连成的边：

1 2 3 4.
5 6 7 8.
1 5.
2 6.
3 7.
4 8.

这个简单的输入数据集合足够画出该立方体的线型模型。但如果要把这个输入数据集合存贮起来以便画出实物体模型的话，采用的输入方法就有差别了。对于实物体模型，不仅要给出立方体所有的6个面，还要以反时针方向给出从物体外部视点看到的物体每个面的4个顶点，因此，我们输入如下数据取代前面输入的6行数据：

```
1  2  3  4.
8  7  6  5.
8  5  1  4.
5  6  2  1.
6  7  3  2.
7  8  4  3.
```

为了搞清楚第二组6行数据的含义，我们必须注意是从物体外部观察物体的，所以物体顶面的顶点方向是从物体正上方观察物体的逆时针方向，对于图1.8和图1.9，即为1 2 3 4。而底面的顶点方向是想象从物体下方观察物体的逆时针方向，即为8 7 6 5，但如果从物体的正上方观察底部，该方向便变成顺时针方向了。

对于顶点的方向，可能发生两种判定错误，一是从物体内部某点观察边界面，而非从物体外部进行观察；二是采用顺时针而非逆时针方向。当同时发生这两种错误时，两种错误便相互抵消了。因此，当我们难以想象从物体外部观察一个面来找出其逆时针方向时，便可以采用错误的面和错误的方向，即从物体内部观察边界面的顺时针方向。之所以要给出边界面顶点的逆时针方向次序，是因为在隐线消除算法中需要它。消除隐线的命令为H，我们将在下节讨论。正如我们看到的，逆时针方向依赖于视点的选取，如果从错误面观察，逆时针方向便变成顺时针方向，反之亦然。对于给定的边界面，程序利用这个事实很快地判定出它相对于视点是否为背面（backface），如果是简单地忽略它，不采用耗时的隐线消除算法。如图1.8和1.9中，相对于相应的视点各有三个背面（其实，D3D仅利用面的前三个顶点来判定该面是否为背面的，但要求第二个顶点必须是凸顶点，这将在1.7节中讨论）。

当只有两个顶点时，点编号的顺序是无所谓的，例如，在

```
F
2  8.
```

中，它对应立方体的两个对角顶点。因此输入两点编号表示由这两点构成的线段。如果输入三个不在同一直线上的三个点，则表示是一个角形。输入四个以上的点，当它们不是一个多边形的顶点，就会带来一些问题。很清楚这些点必须在同一平面内，如果不是，就会出现如下错误信息：

```
Not in the same plane
```

和使用I命令（前节讨论过）一样，在输入边界面时，当按了F键，D3D程序便保持这个状态，如果执行其它命令后再恢复这个状态只需再按一次F键。在该状态下输入一条线时，先输入一个顶点编号，如果操作有效，很清楚它属于I或F命令；如果操作无效，在输入第一个数字时便出现如下错误信息：

```
Don't begin with a digit
```

注意输入每个边界面的最后一个顶点之后都必须插入一个句点（·），原因是边界面可

以是任意多边形，每个多边形可能有多个顶点，这样会出现屏幕上一行放不下一个多边形顶点的情况，为此需要一个标志点数目结束的标记，而Enter键是不行的，所以采用了句点。

当我们输入边界面时，可能发生错误，输入了我们不想输入的边界面，我们可以用D命令删除这个不想要的边界面，而D命令是删除该面包含的点，这会破坏视图，因为删除的点可能是其它边界面所需要的，事实的，这是可能的，删除一个点时会影响这个点上的所有边界面。因此需要采用另外的方法删除一个边界边，这种方法的操作过程是：先按：

```
L
```

其意义是List，即列视图的所有边界面，每列出一个边界面后，便提问OK，例如：

```
1  2  3  4
OK? (Y/N)
```

如果键入N，由顶点 1 2 3 4 构成的边界面被删除，但并不是删除这 4 个顶点，因此不会影响其它相关的边界面；如果回答为y，视图保持不变。这样处理完一个边界面之后，系统将显示下一个边界面，等等。如果想终止这一过程，在响应OK时按Y或N之外的其它任何键即可。

输入一个物体的所有数据之后，例如前例中的一个立方体，物体视图在屏幕上是可见的，现在我们可使用模式变换（mode）命令消点视图中的点编号和三根坐标轴了：

```
M
```

在响应如下提问时键入N：

```
Point number? (Y/N)
Axes?(Y/N)
```

（其实，Y和y之外的任何回答都作为N来处理），如果第二个提问时回答y，那么立即出现如下提问：

```
Aux Lines? (Y/N):
```

这是关于顶点辅助线，即从每个点到xy平面的垂线，它们对于观察三维空间中“丢失”的点是有用的。一般情况下，我们键入N消除辅助线。接着又出现如下问题：

```
Bold Lines? (Y/N):
Entire Screen? (Y/N):
```

如果我们分别回答Y和N，物体重新画成如图1.10所示的视图，其中没有点编号，也没有坐标轴和辅助线，正如我们在 1.2节所提到的，视图的前部包含的线比后部的线要粗些，这比采用同样粗的线更容易说明线型模型，图1.10便是例证。有些情况下，如果你要使视图中所有线都一样粗细，只要在上面bold Lines提示下键入N即可。最后，我们使用整屏（不是在半屏幕）来显示视图，这只要在上面Entire Screen提示下键入y即可，这和下节要讨论的E命令相类似。

值得注意的是缺省模式，即在键入：

```
D3D
```

启动该程序之后以YYYYN顺序分别回答上述 5 个问题，这样屏幕上立刻出现一个完整的物

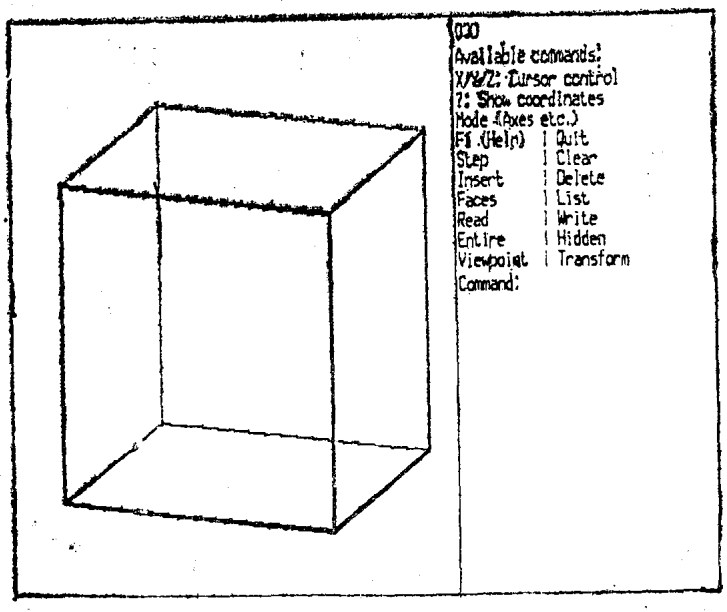


图1.10 立方体

体视图。我们通常采用M命令消除顶点编号，坐标轴和辅助线。因此如果用 1.2 节讨论的R命令从一个文件中读取一个完整物体数据时，这种缺省模式就需改变了，除非只从文件中读取点（没有任何面或线段），这时点编号和轴都是不可见的。因此命令R 改变了缺省模式。但是我们可以使用命令M恢复这些点编号和轴等。

1.5 整屏、隐藏线和打印

现在要在矩阵打印机上打印出图形结果，在此之前我们要使用整屏命令放大透视图和消除信息区域，整屏命令为：

E

在上一节我们看到，可以使用模式命令（M）来进行整屏，但在使用该命令要回答一系列的问题，如显示或消除点编号、坐标轴、辅助线和线的粗细等。如果不想改变这些选择，那么可直接使用E命令，该命令避免了一系列的提问。到现在为止，进入整屏状态可采用两个命令M和E，它们达到同样的效果。

当在打印机上打印出图形时，选择适当的比率是很重要的，这里的比率指的是横竖尺寸之间的比率。例如，屏幕上生成的一个正方形要准确在打印机上打印出来，不是打印成长方形或梭形；屏幕上的圆不是打印成椭圆。因为该程序简单地把屏幕象素映射成打印机上的点。屏幕上的象素在打印机上打印时可以通过如下命令自动进行修正：

A

它在E命令之后使用，键入A后，可能破坏屏幕上的图形，但当你连通打印机，并键入：

P

打印机以正确的尺寸打印这个图形。如果你更喜欢屏幕上的比例（例如不打印屏幕图形），那么你可以按A之外的任何键。记住这两种方式不能同时存在，即调整屏幕上的图形而不调

整打印机上打印出来的图形比率，或者调整后者而不调整前者。

不用命令A，可以使用O命令把图形写到一个输出文件中，它主要是用作其它程序，如程序PLOTHP的输入文件，以便在HP绘图机上把该图形绘制出来，这将在 4.6节讨论。在本书中你看到，一个在矩阵打印机上打印很粗糙的图形，在HP绘图机上绘制得很漂亮。最后你要记住，D3D的输入文件和使用O命令输出的文件在格式上是不同的（尽管都是ASCII码文件）。为了避免混淆，我们称前者为物体文件，扩展名采用.DAT，后者称为绘图文件，扩展名采用.PLT。

一个图形作为P命令的结果在打印机上打印出来之后，或者不想打印而直接按Enter键，透视图均消失，屏幕恢复成使用E命令之前的样子。

通常，命令E增大了屏幕上可见的图形。同时放大了图形中的点编号和坐标轴，但并不增大图形的纵横比率，如果图形中存在一些最终结果不想要的东西，你可以使用M命令而不是E命令来消除它们，这些在 1.4讨论过。除了可以改变模式外，你还可以改变视点，这要使用 1.2节讨论过的V命令。

如图1.11(a)和(b)所示，它们分别对应E命令的前后屏幕形式，其中的图形是一个L形物体图形，采用线型模型显示。

在大多数应用中，我们更喜欢实物体模型，换句话说，要消除隐线，程序D3D可以这样做。当该物体显示成线型模型时，键入：

H

它是隐线消除（hidden_line removal）的简写，此时屏幕上出现如下提问：

Do you want the object to appromate curved surfaces? (Y/N):

这时我们作否定回答：

N

然后我们可以选择如下两选项之一：

A：与命令p组合使用把结果输出到矩阵打印机上

O：获取一个绘图文件

如果这两者都不选择，则可按其它任何键。这之后我们可以按命令p把结果送给矩阵打印机。对于曲面，命令H类似于命令E，同样，键入A便调整打印图形的纵横比率，使用O得到一个绘图文件。

现在讨论隐线消除过程，对于复杂的物体（使用带8088处理器的简易PC）消除隐线可能要花很长时间。例如前面讨论过的例子，我们获取图1.11(c)，其中除了包含完全可见的边和完全不可见的边外，还有一条部分可见的边。改变模式命令M不能应用于H命令，隐线消除总是使用整屏，并且所有线都同样粗细，当返回正常工作屏幕之后，重新出现有关信息，恢复原来的模式。

实际上，图11.1(c)是使用命令O和程序 PLOTHP（见4.5节）在HP绘图机上绘制出来的，而图1.11(a)和1.11(b)则是在矩阵打印机（Star NL10）上打印出来的。我们不仅可以在命令E或H之后，还可以直接在工作屏幕状态下使用命令H，图1.11(a)便是保证。在绘图机上通过PLOTHP绘制的图既不包含点编号，也不显示出线的各种粗细，这不一定满足不同用户的需要。因此，如果你想绘制如图1.11(c)那样的图，可使用绘图机，否则使用打印