用 PASCAL 实现三维图形

晏海华　编著

学苑出版社

计算机程序设计语言系列丛书

用 PASCAL 实现三维图形

晏海华　　编著

学苑出版社

1993

(京)新登字 151 号

内 容 提 要

本书以 PASCAL 语言为例，深入浅出地介绍了图形技术的基本概念和方法。读者可在本书中找到关于二维、三维图形的各种旋转、移动、显示等方面的巧妙的算法并有相应的 PASCAL 程序实现。内容翔实、生动、图文并茂、易学易用。对初涉计算机图形学的科研人员具有很强的参考价值。

欲购本书的用户，可直接与北京 8721 信箱联系，邮码：100080，电话：2562329

计算机程序设计语言系列丛书

用 PASCAL 实现三维图形

编　　著：晏海华
审　　校：燕卫华
责任编辑：徐建军
出版发行：学苑出版社　　邮政编码：100032
社　　址：北京市西城区成方街 33 号
印　　刷：北京四季青双青印刷厂
开　　本：787×1092 1/16
印　　张：8.125　　字数：182 千字
印　　数：1—5000 册
版　　次：1993 年 9 月北京第 1 版第 1 次
ISBN$_7$—5077—0807—1/TP・18
本册定价：7.00 元

编 者 的 话

图形技术是当今计算机发展的热门课题之一，越来越多的人被吸引到这个引人注目的领域从事研究．当我们从计算机屏幕上看到那栩栩如生的三维图形和动画时，是否觉得它太神秘而难于实现呢?

本书以 Pascal 语言为例，深入浅出地介绍了图形技术的基本概念和方法，您可在书中找到关于二维(平面)、三维(立体)图形的各种旋转、移动、显示等方面的巧妙的算法并有相应的 Pascal 程序实现，而且内容翔实、生动、图文并茂、易学易用．对初涉计算机图形学的科研人员来说是一本不可多得的实用教材．

编者

一九九三年四月

目 录

第一章　概述

1.1　什么是计算机图形

计算机图形软件的第一次实现是在 60 年代初(由 D.E.Sutherland, P.Bezier 和 P.de.Casteljau 实现)。当时，由于计算的复杂性极大地限制了计算机图形的应用，使它仅用于工程目的如汽车和飞机部件的设计。由于后来计算机技术的飞速发展，计算机图形学开始应用于其它领域。从八十年代开始，计算机图形迈进了微机领域。许多可视游戏，往往具有复杂的图形，就是一个很好的例子。

今天，计算机图形主要应用在以下几个方面:

• CAD / CAE(计算机辅助设计 / 计算机辅助工程): 技术图的交互设计，如二维图(构造平面图、VLSI 设计等)和三维图(如在建筑和机械工程中的透视图等);

• 制图学;

• 模拟: "交互式动画片"(如用于训练飞行员的飞行模拟器、计算机游戏);

• 用于科学目的交互式三维空间构造(如大分子结构的检查);

• 计算机生成娱乐电影(如卡通片和电视片头，以及超长动画片等);

• 用计算机输出的图形来形象地表示数据。例如可在电视上用图形显示选举结果。在微机上同样能用图形显示和交互图形模式表示用户界面(如使用鼠标操作窗口和键入命令等)。

1.2　本书内容

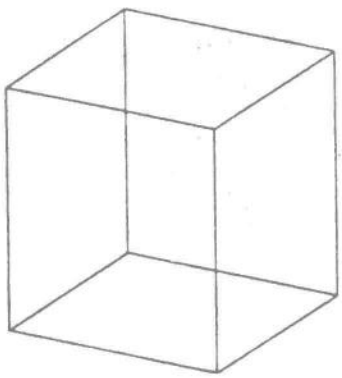

图 1

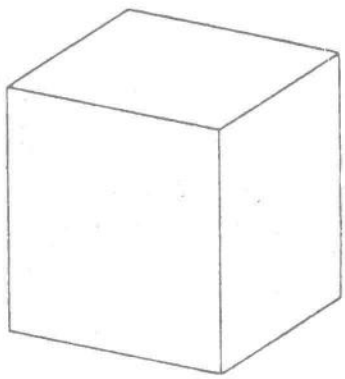

图 2

本书主要讨论用于三维对象的计算机图形表示方法。第二章和第三章介绍某些数学基础，数据结构和计算机中处理几何对象的过程。第四章说明在一个平面上(屏幕)表示三维

对象的各种方法．用于产生某些有趣的三维景象的过程将在第五章讨论．这些例子用来进一步说明所介绍的方法．在前五章中的这些例子和程序不仅能够用来描述图 1　这样的简单图形，而且能够画图 3　这样相当复杂的图形．

图 1 为一个立方体的平行投影，即一个透明的立方体，所有的边都可见，也包括纸盒的那些实际上看不到的边，因为它们在"背面"．许多常见的方法是去掉这些"隐藏的边"来描述实际的图象，对于立方体来说，这可能没有必要．但跟图 1　相比，图 2 中的立方体可能更形象．换句话说，图 3 实际为一堆杂乱的线．如果去掉隐藏线，这种三维对象的结构将变的更加清楚．(如图 4)

在第六章中，我们通过简单对象如立方体等(如图 2)来说明去掉隐藏线的方法．然后，针对更加复杂的结构，介绍二种不同的去掉隐藏线的方法．将可使得相当复杂的图形对象更加清晰．在第六章结束，将综述其它几种去隐藏线的方法．

在第七章中，将介绍某些应用，以及进一步扩充已有的图形程序．首先，将讨论象三维直方图这样的实际应用．然后，将进一步说明，计算机图形通过使数学(解析几何)与计算机科学相结合，能增进学校中的计算机科学的教学．

本书的主要目的是展示计算机图形学中很有意义的部分．即用线来画三维对象，并尽可能使得同实际看到的一样．对个别方法，不仅从理论上加以介绍，而且通过详细的程序例子加以演示．

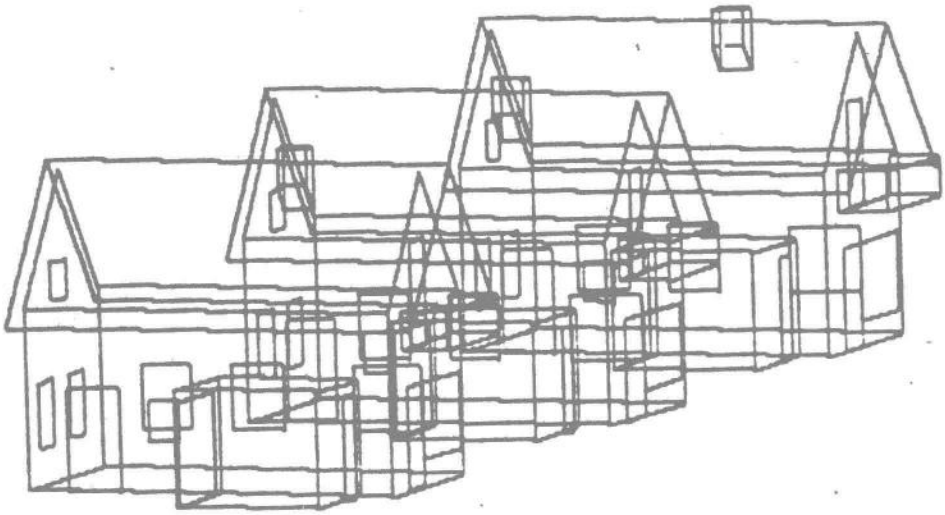

图 3

当然，本书不可能描述整个计算机图形学．然而，在最后一章，我们将给出计算机图形学的某些方面的概要，这些都是本书中没有提到，但我们认为是很重要的．这为进一步的研究提供了选择．

1.3　必要知识

在本书所讨论的图形技术中，并不需要深奥的数学知识．然而某些解析几何的基础是需要的．如，一个面向计算机的立方体描述(如顶点坐标的表示)，在我们能在计算机上画它之前，解析方法是需要的．在某些例子中(如图 49)，一个多边体的边要穿过其它多边体．为生成这样的图象，显然我们必须知道该如何计算线和面的交叉．同样，描述一个对象在空间的移动(旋转，平移)的转换是非常有用的，它使得我们能从不同的角度来观察对象．

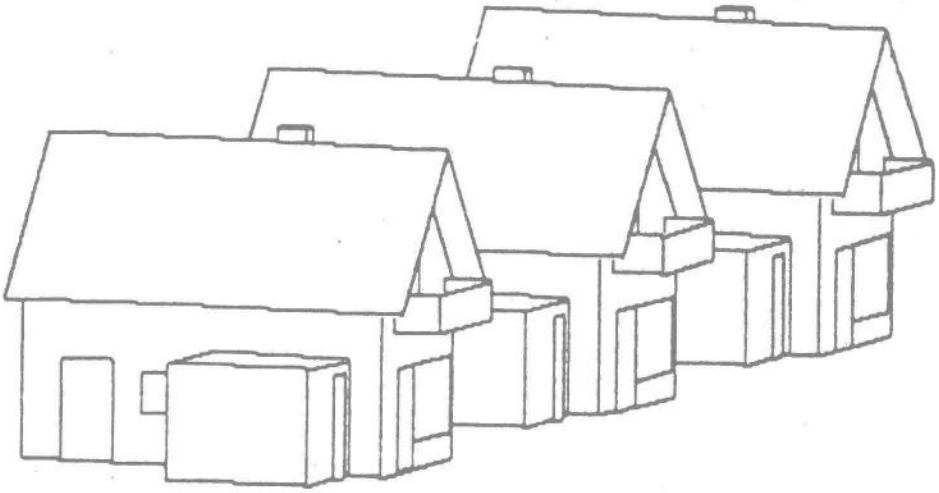

图 4

在 2.1，2.3，3.1 和 3.3 节中，我们挑选该类型的某些基本实例．各种方法在此是通过例子来说明的，而没有证明．具有某些解析几何知识的，或只对使用本书中程序感兴趣，而不关心这些方法是如何工作的细节的读者，在首次阅读时，可以忽略这些章节．

1.4 PASCAL 程序

本书的主要部分是由 PASCAL 过程组成，用以实现图形算法．它们可以用各种方式结合起来，演示这些算法是怎样工作的．在下面章节中，将通过一定量的例子来说明．此外，这些过程当然可以用在用户自己的图形应用中．由于 PASCAL 程序是模块结构，这意味着并不必知道每个过程的细节．即它们可以看成是一个"黑盒"，我们只需要知道它们所要求的输入，以及产生的结果．

这些程序已经在运行 UCSD 和 TURBO-PASCAL 的不同微机上测试过．在此，已为 APPLEII，APPLEII+，APPLEIIe(UCSD-PASCAL)，以及 IBM-PC 和兼容机(TURBO-PASCAL)准备了这些程序的盘．然而，我们写程序时，已经意识到避免使用某些 PASCAL 的方言，或依赖于具体机器的图形界面．因此，在此所介绍的图形程序可在任何能为 PASCAL 提供高分辨率图形的机器上运行．但是，有两件事情必须要做：

1. 调用 PASCAL 系统过程来画一个线段，其必须放在过程 Draw 中(见 2.4 节

)。

2. 用以描述屏幕的常量必须正确(见 2.4 节)。

除了模块化外，选用 PASCAL 程序设计语言的另一个理由是：PASCAL 的编译程序几乎在所有的微机系统上都有。因此，在很大程度上，我们可以用 PASCAL 代替 BASIC(其缺点是众所周知的)来进行图形处理。然而，从原理上来讲，将程序转换为其它程序设计语言是可行的。在此，为了方便，我们没有使用 PASCAL 中的指针变量，尽管在动态存贮分配中使用指针具有优越性。但我们没有放弃使用递归程序，因为它能极大地简化某些程序。

在本书中，我们并没有在程序中使用特殊的外部设备，如图形数字化仪，鼠标或绘图仪。这些程序必须经过适当的修改才能适应相应的设备，但是，简单地数据结构使得扩充图形包以适应特定的设备配置是容易的。例如，将输出设备改为绘图仪。我们可以简单的象其它 PASCAL 系统一样使用这些程序。

我们已做到尽可能简单而清晰的安排程序。为止，我们并没有优化它们的运行时的时间和空间。在 7.1 节，我们给出了优化程序的某些提示。

第二章　二维(2D)图形

在本章中，我们将讨论某些二维图形的概念，并为后面的三维图形做准备。

在 2.1 节介绍了图形映象的基本构件。在 2.2 节中，说明了为处理二维图形对象所用到的一些数据结构和过程。在 2.3 节中，探讨了在平面上转换几何对象，如旋转，放大缩小，以及平移等的不同方法。最后，我们介绍将一个图形的一部分映射到显示屏上，并"裁剪"不能放入屏幕的部分的方法。这将在 2.4 节讨论。

我们并不证明在 2.1 节和 2.3 节中所介绍的方法，但通过某些简单的具体例子来说明它们是正确的。在 3.1 和 3.3 节也采用了同样方法。我们希望这能给那些在学校已经学过解析几何知识的人温故知新，同时也为那些首次遇到的人做出一些有用的介绍。更为详细的介绍请参考我们所推荐的参考书[10]。

2.1　向量，点，线段和多边形。

为了能定量的处理几何对象，它们需要正确的描述。为此我们使用成直角的 X 轴和 Y 轴的笛卡尔坐标系统。一个平面上的点可用其相应的 X 和 Y 坐标来唯一表示，即用一对数字(X，Y)表示。例如，图 5 中的两个点 a 和 b 可表示为 a＝(1，2)和 b＝(5，3)。

圆括号括起的一对数字(X，Y)也称为一个(二维)向量。我们可以认为是一个从原点(0，0)开始，(X，Y)结束的矢(如图 6)。在实际中，我们并没有区分一个点(X，Y)和向量(X，Y)。在下面，我们使用黑体小写字母来表示它们，如 P＝(X，Y)。

把点看作向量的优点是向量可以相加，也就是说两个向量 a 和 b 可以结合成一个新向量 a+b。准确的说，向量 a+b 是按下面方式得到的：平移向量 b，———即不改变其方向和长度———使其起点正好为向量 a 的末点。这样，向量 a+b 为一个以向量 a 的起点为起点，以平移向量 b 的终点为终点的向量(见图 7)。

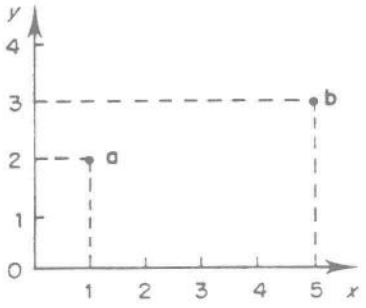

图 5

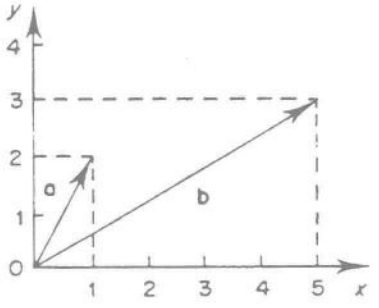

图 6

向量 a 和平移后的向量 b 组成了一个平行四边形，而向量 a+b 为其对角线。这个平行四边形通常称作"力的平行四边形"。该命名源于物理中对向量的应用，即用向量表示

力．而向量的长度表示力的大小，其方向表示力的作用方向．而两个力作用的结果即为两个相应向量的和．根据力的平行四边形原理，我们可以得到：a+b＝b+a．

两个向量的和计算起来非常简单，只需将相应的 X 和 Y 坐标相加即可．见图 7．

a+b＝(1，2)+(5，3)＝(1+5，2+3)＝(6，5)

类似地，我们可以用一个数字乘以一个向量．即把该向量的每个坐标乘以该数字．例如：

0.5 * b ＝0.5 * (5，3)＝(0.5 * 5，0.5 * 3)＝(2.5，1．5)

(在 PASCAL 中，我们通常用十进制实数描述点，而向量对应的两个坐标值用逗号隔开．)

用数字乘以向量意味着结果向量同原有向量方向相同，但其长度伸长或缩短了所乘数字倍．在上例中，向量 b 被 0.5 相乘．其结果向量的长度为原有的一半．

如果一个向量同一个负数相乘，其结果向量将反向(180 度)．例如同−1 相乘，将导致向量旋转 180 度，而长度不变．这样，我们可以表示两个向量相减，如 b−a，即为向量 a 乘−1，然后同 b 相加．

b−a＝b+(−1) * a＝(5，3)+(−1，−2)＝(5−1，3−2)＝(4，1)

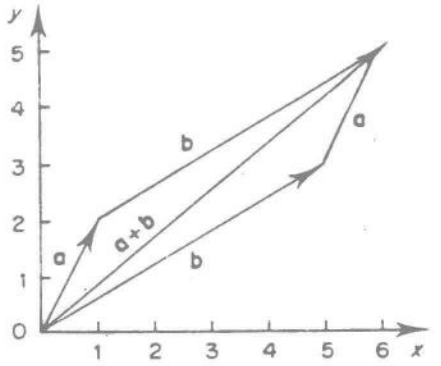

图 7

图 8 显示向量差 b−a，首先，从坐标系统原点画出 a 和 b 向量，然后平移连接 a 和 b 结束点的向量至原点即得向量差．从图上可以看出不同的向量是怎样构造的．

我们使用向量差给出一个通过点 a 和 b 的线 g 的简单描述．向量 b−a 指定了该线的方向，因此我们又称它为 g 的方向向量(direction vector)．g 上的每个点可以描述为向量 a 加上方向向量乘以一个数 t(即可伸长或缩短，甚至当 t 为负数时可以使它反向)．当 t＝0

时，即为点 a，而当 t＝1 时，即为点 b。图 9 同时还显示了当 t＝−0.5 和 t＝0.5 时的点的向量。

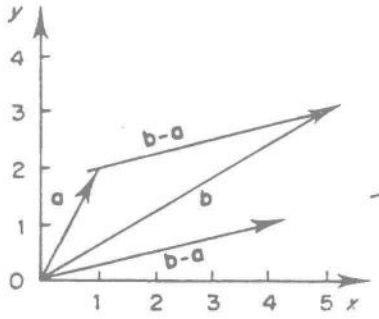

图 8

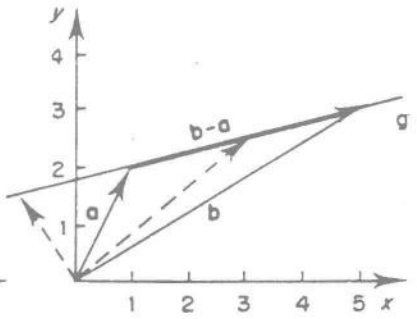

图 9

通过下式，由点 a 和 b 组成了线 g 上的所有点 P：

$$P = a + t * (b - a)$$

此时，t 为一个数(实数)，我们称它为点 P 的参数。在此，线 g 的这种表示我们称为 g 的参量表示。

对于用参量化表示的线 g 和 h，我们可以按下面的方法计算它们的交点。例如，线 g 即为图 9 中所显示的，而线 h 为通过点 C＝(1，4)和 d＝(9，−2)的线。相应的线 g 和线 h 的参量表示为：$P = a + t * (b - a)$和 $P = c + s * (d - c)$。线 h 的参数为 S 以区别于 t。一个同时在两条线上的点应同时满足这两个表示，因此可以写出下面等式：

$$a + t * (b - a) = c + s * (d - c)$$

或，将相应数字代入

$$(1，2) + t * (4，1) = (1，4) + s * (8，-6)$$

按坐标分开的话，该向量等式可分成两个等式：

$$4t - 8s = 0$$

$$t + 6s = 2$$

从这两个等式，我们得到 t＝0.5，s＝0.25。将 t 和 s 的值代入相应参量表示，产生公共交点为(3，2.5)。(试着用该方法计算一下两条平行线的交点——它是不存在的)。

在我们图形应用中，屏幕只是平面上一个非常小的区域。因此，一条线(无限长)永远不能完全显示出来。所以，点 a 和 b 之间的线段，我们记为[a，b]，这正是计算机图形学中所关心的。而不是通过这些点的线。

从上面 g 的参量表示，我们可以容易的得到[a，b]的参量表示法。a 和 b 之间的线段由参数值为 0 到 1 的线 g 上的点组成。参数值 t＝0 和 t＝1 分别对应线段[a，b]起始点 a 和终点 b。而 t＝0.5 对应其中点(见图 9)。

线段上的某一点 P，其等式可稍微如下改变：

P = a+t * (b−a) = a−t * a+t * b = (1−t)a+t * b

注意：这样的"混合计算"-----一方面有向量 a，b，另一方面有数字 t-----它服从使用数字进行普通计算的同样规定。这是可以接受的，因为向量计算仅包含普通数字计算，只不过两个坐标是分别计算的。

在简短介绍了向量计算后，一个线段的参量表示可归结如下：

点 a 和 b 之间的线段[a，b]，其上所有点满足：

P = (1−t) * a+t * b

其中 0≤t≤1.

我们已经在好几处提到向量 a 的长度，现在，我们将计算向量 a = (x，y)的长度。如图 6 所示，坐标系统的原点，向量的终点，以及点(x，0)构成了一个直角三角形。它的两个直角边长分别为 x 和 y，而其斜边的长度即为向量 a 的长度。因此，根据勾股定理我们得到：

向量 a = (x，y)的长度为$|a| = \sqrt[2]{(x^2 + y^2)}$。

两个向量之间的角度可由数积(Scalar product)得到。

两个向量 a = (a1，a2)和 b = (b1，b2)数积(或内积)<a，b>定义为：

<a，b> = a1 * b1+a2 * b2

因此，两个向量的数积是一个数字(而不是一个向量)。它为两个向量 X 轴坐标的积同 Y 轴坐标的积的和。由图 6 中向量，我们有：

<a，b> = <(1，2)，(5，3)> = 1 * 5+2 * 3 = 11

数积同两个向量之间的角度 α 有关，α 可由下面公式求得(可由余弦定理证明)。

$$\cos\alpha = \frac{\langle \mathbf{a},\mathbf{b} \rangle}{|\mathbf{a}| * |\mathbf{b}|}$$

在图 6 中，向量 a 的长度为$|a| = \sqrt[2]{(1^2 + 2^2)} = \sqrt[2]{5}$，b 的长度$|b| = \sqrt[2]{(5^2 + 3^2)} = \sqrt[2]{34}$，因此$cos\alpha = 11 / \sqrt[2]{5 \times 34} = 0.8436615$，$\alpha = 32.47^0$。

实际上，数积还可用来检查两个向量是否正交等，即它们之间的角度为 90 度。此时，该角度的余弦值为 0。

如果两个向量的数积为 0，则它们正交(orthogonal)

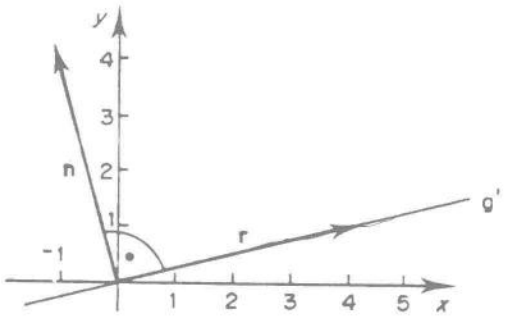

图 10

例如，图 10 中向量 r=(4，1)，n=(−1，4)。正交向量的概念提供了一个不用参量表示描述垂直线的方法(有时是非常有用的)。在图 10 中，线 g′为通过原点，并与向量 r 同方向的线。而 g′上的点的参量表示为 p′=t*r，t 为实数。此外，线 g′上的点我们也可以通过正交于 r 的向量 n 来描述。一个点 p′在 g′上(可解释为向量 p′)，只有当它同 n 正交，即<p，n>=0。

假定 g′为通过原点，并与向量 r 同方向的线。如果向量 n 同 r 正交，则 g′上的任一点的向量 p′同 n 正交，即<p′，n>=0。

这叫 g′的正交表示。向量 n 称为 g′的正交向量。

上面线的正交表示还有一个明显缺陷。即只能表示通过坐标原点的线，而不适合于其它线，例如，图 9 中的线 g。该线平行于 g′(即它们有相同的方向向量 r=b−a)，而 g 上的点 p 可通过向量 a 加上 g′上的点 p′获得，即 p=p′+a 。因此对于 g 上的点 p 有：

<p，n>=<p′+a，n>=<p′，n>+<a+n>=0+<a，n>

(其中第二个等式可通过数积的特性或通过具体数字代入得到验证。)

下面，我们得到一条任意线的正交表示。

一条通过点 a 和 b 的线上的点 p，可表示为：

<p，n>=<a，n>

n 同线 g 的方向向量 b−a 正交，即<n，b−a>=0。

如果 r=(rx，ry)是线 g 的一个方向向量，并且 n 正交于 r，因此 n 为 g 的正交向量。例如 n=(−ry，rx)就是其中一条正交向量。因此由参量表示，我们很容易得到正交形式。例如，图 9 中的线 g，其上的点 p=(x，y)满足等式<(x，y)，(−1，4)>=<(1，2)，(−1，4)，即−x+4y=7(将数积展开后得到)。

由线的正交形式，我们可以计算两条线的交点，----这类似于参量表示的情况，---通过建立含有两个变量的等式得到。如果第二条线通过点 c，并且其正交向量为 m，则有下面两个等式：

<p，n>=<a，n>和<p，m>=<c，m>

其中两个变量 x，y 为交点 p 的坐标，即 p=<x，y>。

一条线将平面分成两个相连的平面，通常称为半面(half−planes)。假如线的正交形式为<p，n>=<a，n>，则两个半面上的点 p 分别满足两个不等式：<p，n><<a，n>和<p，n>><a，n>。如图 9 中，一个点(x，y)如果满足不等式−x+4y>7，则其在线 g 的上面，满足−x+4y<7，则其在线 g 的下面。一般，不等式，'<'将半面描述为正交向量上的点(如图 10)，而不等式'<'说明另一半面。我们将图 9 中描述线 g 的正交向量(1，−4)改为(−1，4)就可以了解到这点。注意：如果不等式同一个负数相乘，不等式要反向(即'<'改为'>')。

当使用半面时，通过不等式系统，我们可以了解平面区域特点。例如，图 11 中的斜线部分的点(x，y)满足，X 轴上，Y 轴右边，g 和 h 线下。因此，我们可以得到下面 4 个

不等式(它们必须同时满足)。

$x \geq 0$

$y \geq 0$

$-x+4y \leq 7$

$6x+8y \leq 38$

一个有限的半面交叉得到的平面区域我们称为(凸面)多边形。凸面意味着多边形中的任何两个点 a 和 b，而线段[a，b]同样是多边形的一部分。见图 11，这是显然的。相对应，在图 12 中，表示的是一个非凸面多边形。在后面我们将看到，凸面多边形在图形应用中非常普便(同非凸面多边形相比，计算起来简单)。半面相交是一个凸面，这是由于每个半面本身就是一个凸面，而平面的凸面区域相交同样是凸面。点，直线段和多边形是本书图形的三个基本元素。

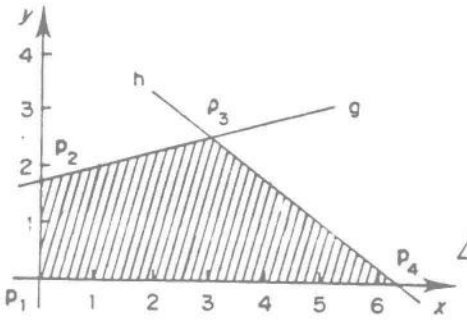

图 11

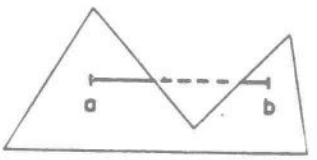

图 12

在图形应用中，通常一个多边形 P 并不是由前面的不等式来定义，而是由其顶点 P_1，...P_n 来定义。一个多边形的边界由几个线段$[P_i, P_{i+1}]$，$i=1$，...$n-1$ 和$[P_n, P_1]$组成。一个由不等式描述的多边形我们可以得到其边界描述，正如我们由线的参量表示得到其正交表示一样。向量 $p_{i+1}-p_i$ 是线段$[p_i, p_{i+1}]$的方向向量。假定点 p_i 的坐标为(x_i, y_i)，则向量 $n_i=(y_i-y_{i+1}, x_{i+1}-x_i)$垂直于 $p_{i+1}-p_i$。假设，多边形的顶点是顺时针排放的(正如本书所表示的)，因此，该向量指向多边形外面。它称为外正交向量。因此，一个多边形内的点(x，y)，它满足下面 n 个不等式：

$$<(x, y), n_i> \leq <p_i, n_i>, \quad i=1, \ldots n$$

(这里，"最后一条边"$[P_n, P_1]$的正交向量定义为 $n=(y_n-y_1, x_1-x_n)$)。按上述方式我们得到图 11 的四边形($n=4$)的不等式，它似乎不同于前面所得到的该四边形的不等式。第一个不等式被乘以 $-4/7$，第二个被乘以 $4/3$，第三个被乘以 $12/5$，第四个被乘以

-3 / 19. 同前面描述图 11 的四个不等式相比，只有次序上的不同(显然这是无关紧要的).

可能本节所描述的似乎有些理论的和乏味的. 对于那些从来没有接触过向量计算和解析几何的人来说，它可能不容易理解. 然而，它能增进对本书为平面几何，以及三维几何所开发的数学工具的理解. 本节所介绍的技术只需稍加修改就能适应三维空间，这将在 3.1 节中讨论.

2.2 二维图形的数据结构.

现在，我们开始进行计算机图形实践. 首先，我们介绍程序包的数据结构以及过程. 本书中，最基本的元素是点和连接点的线段. 点和线段分别存贮在数组 D2Pt 和 Segm 中. 一个点由其坐标 x 和 y 来表示，而线段[b，e]通过存贮在数组 D2Pt 中的起始点 b 和终止点 e 来表示. 我们需要用到下面的常量，类型以及全程变量，我们将解释它们的作用. 它们是程序包 D2Pac1 中的一部分，该程序包中还包括本章所有作了相应注释的过程和函数.

```
const MaxD2PtNumb=500;
    MaxSegmNumb=500;
    ScreenWidth=711;  { dimension of screen }
    ScreenHeight=349; { with Hercules card for IBM-PC }
    TurnRound=false;  { =true, if (0,0) is upper left corner }
    AspectRatio=0.69;


Type  PtType=array [1..3] of integer;
    VectType=array [1..3] of real;
    SegmType= record b,e,colour:integer end;
    MatType=array [1..3] of VectType; { matrix as array of row vectors }

var   D2Pt:array [1..MaxD2PtNumb] of PtType;
    D2PtNumb:integer;
    Segm:array [1..MaxSegmNumb] of SegmType;
    SegmNumb:integer;
    WindowLe,WindowRi,WindowUp,WindowLo:integer;
    ViewportLe,ViewportRi,ViewportUp,ViewportLo:integer;
    xmin,ymin,xmax,ymax:integer;
```

在此，一个点不是存贮在长度为 2(对应其 x，y 坐标)的数组中，而是存贮在长度为 3 的数组中，这主要为后面的三维图形考虑，其需要另一个坐标 z 来描述一个空间中的点. 在本章中，所有的 z 坐标都设置为 0. 在定义线段时，除了需要其开始和结束点坐标外，还要存贮其色彩. 该信息用来在彩色屏幕上或多色绘图仪上产生彩色图形. 常量 MaxD2PtNumb 和 MaxSegmNumb 定义了数组 D2Pt 和 Segm 的维数(即长度). 变量 D2PtNumb 和 SegmNumb 指示最后存贮的点或线段在相应数组 D2Pt 或 Segm 中的位

置。(因为，这两个数组并不是全都使用)。

下面三个过程用来初始化数组 D2Pt 和 Segm，以及初始时的点和线段。

```
  D2PtNumb:=0; SegmNumb:=0;
  xmin:=32000; ymin:=32000; xmax:=-32000; ymax:=-32000;
end; { D2Clear }

procedure D2PtIn(x,y,z:real); { D2Pac1/D2Pac2 }

{ enters (x,y,z) in D2Pt , updates xmin, ymin, xmax, ymax }

begin
  D2PtNumb:=D2PtNumb+1;
  D2Pt[D2PtNumb,1]:=round(x); D2Pt[D2PtNumb,2]:=round(y);
  D2Pt[D2PtNumb,3]:=round(z);
  if x<xmin then xmin:=round(x); if x>xmax then xmax:=round(x);
  if y<ymin then ymin:=round(y); if y>ymax then ymax:=round(y);
end; { D2PtIn }

procedure SegmIn(b,e,colour:integer); { D2Pac1 }

{ enters [b,e] in segment list }

begin
  SegmNumb:=SegmNumb+1;
  Segm[SegmNumb].b:=b; Segm[SegmNumb].e:=e;
  Segm[SegmNumb].colour:=colour;
end; { SegmIn }
```

在过程 D2Clear 和 D2PtIn 中，我们对变量 Xmin，Ymin，Xmax，Ymax 进行赋值，它们分别表示所有点 x，y 坐标的最小和最大值。这些信息对于在屏幕上以适当比例显示图形是必须的。在 2.4 节，我们将详细说明其它常量的作用。

为使程序清晰，我们在程序中使用了大小字母。大多数 PASCAL 编译程序(至少是我们所知道的)并不区分大小写字母。因此，我们在程序中使用大小字母，并不改变程序的含义。

在下面的第一个小应用中，我们写了一个画大象的程序，如图 13。该程序的基本是过程 Eleph1，它使用前面所介绍的二维图形结构，来存贮组成大象的点和线段的集合。