第1章 图形编程接口介绍

OS/2® 2.0 显示管理程序™(PM)图形编程接口(GPI)提供了200多个函数。这些函数可以使用户用各种技术、媒体和格式在PC环境中建立、显示、打印图形图象。

GPI的一个主要优点就是设备独立。设备独立是通过把设备描述表与显示空间相连，并通过使用多种图形图元来实现的。

本章描述了操作系统的部分内容，这部分内容可以只利用显示空间-设备描述表这对工具就能编写PM应用程序而不必与设备驱动程序直接通讯。下面概述如下主题：

- 显示空间与设备描述表
- 图形图元
  - 一线和弧图元
  - 一面积域图元与多边形
  - 一字符串图元和字模
  - 一标记图元
- 颜色与混合属性
- 位图与元文件
- 路径和区域
- 保留的和非保留的图形
- 关联
- 剪取和边界判定
- 坐标空间和变换操作与函数
- 提交打印作业和操作

以下各章的内容深入地叙述了GPI的组成部分、功能及优点，这些描述有助于编写用于显示和打印输出图形的应用程序。

1.1 关于GPI

在一个PM应用程序中，可用图形函数(它们的大多数都冠以前缀Gpi)来编写所希望的图形输出程序。本书描述了很多GPI函数。要了解完整的、按字母顺序排列的GPI函数表，以及它们的语法和参数，请参考《显示管理程序编程参考手册》。实际上，所有GPI

的调用都符合 IBM 系统应用体系结构"(SAA")的规定。

所编写的应用程序可能会接收到有关将要建立的图象的窗口消息;那么此时的任务就是编写程序如何把该输入信息加工成输出图形。有关窗口消息、PM 用户接口的可视部分、建立和使用窗口编程函数等的细节,请参考 OS/2 编程指南第 1 卷,这样便能生成基于窗口的图形应用程序。

1.1.1 显示空间与设备描述表

显示空间是一数据结构——相当于编程用的一小块空白纸。在图形图象被送往输出设备之前,先建立在该纸上。输出设备可以是:打印机或绘图仪、内存位图、显示屏、甚至是传真卡。每个 PM 应用程序必须把所有图形都送到显示空间。

图形工具和设备驱动程序负责从显示空间获取输出图形送到输出或硬拷贝设备上。从显示空间到硬拷贝设备的映射由设备描述表处理。该设备描述表是与显示空间连在一起的。

设备描述表也是一数据结构——其目的是把给图形显示空间所发的图形命令翻译成为物理设备能够用以转换成显示信息的命令。图 1-1 表示应用程序的图形命令如何通过显示空间流到设备描述表及物理设备。

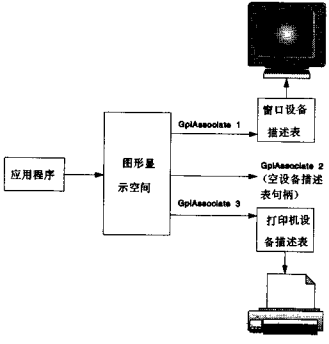

图 1-1 应用程序的图形命令流

1.1.2 图形图元

图形图元是由构造图画的 PM 编程接口所提供的基本组成部分。有几种类型的图元：

- 直线和弧
- 面域和多边形
- 字符串
- 标记

1.1.2.1 图元属性

每个图形图元都有一套属性，这些属性进一步确定了它的外形。例如，一条直线可以用不同的颜色、不同的宽度来画；它可以是点线、实线、甚至是不可见的。这样的属性称为图元属性。

一般而言，图元属性是为图元的类型而不是为图元的某个实例而设的。比如说，如果线类型属性置为 LINETYPE DOT，则其后所画的所有的线都会是点线。

所有的图元属性都有一组缺省值，当显示空间第一次建立时则使用缺省值，若不对这些缺省值修改的话，则这些缺省值一直有效。当一个属性值设置好之后，该值便称为当前属性值，若不改变则一直有效。

1.1.2.2 直线与弧图元

象棋饼图和直方图这样的简单画图，应用程序就是用直线和弧图元作为它们的绘制工具。计算机辅助设计(CAD)应用程序就是用由直线与弧图元组成的许多直线和曲线类型和形式来绘制更复杂的图。

直线和弧图元属性存放在 LINEBUNDLE 数据结构中。其相关的属性包括宽度、类型、颜色和混合。

关于直线和弧图元及其在 PM 应用程序中具体的使用方法，请看第 3 章：线与弧图元。

1.1.2.3 面域图元

面域是用直线和弧图元定义的封闭形状；面域图元是由定义和绘制面域的函数产生的。

面域图元放在 AREABUNDLE 数据结构中。面域图元属性包括模式符号和参考点、模式集、前景和背景颜色、及前景和背景混合模式。

面域也被称作面域括号，因为建立和定义一块面域的函数总是由函数 GpiBeginArea 和 GpiEndArea 括起来。而积括号的好处在于它们能够十分灵活地画出各种各样的曲线或直线并能把其结果组合成一封闭图形。

1.1.2.4 多边形图元

多边形是以直线为界的封闭平面图形。多边形图元是能够填充的或既能填充又能画

轮廓的有规定顶点的多边形。

GPI 有一个函数(GpiPolygons)可用作在面域括号之外画由多条直线边构成的封闭面域;但是与面域一样,这些封闭面积可以相邻、相交或者完全分离。

关于在 PM 的应用中怎样,以及何时使用面域图元和多边形,请看第 5 章"面域和多边形图元"。

1.1.2.5 字符串图元和字体

字符串图元用作建立文本。字体是一组具有共同高度、线宽度和外形的字符串图元。

字符串图元函数和字体函数都用以确定现有字体,为文本输出选择字体,画一串选定的字符,及修改字符串。

字符串图元存放在 CHARBUNDLE 数据结构中。字符串图元属性包括:集合、精度、单元、角度、方向、文本对准、颜色及混合。

关于字符串图元的详细描述请参考第 6 章"字符串图元",关于字体的详细情况请参考第 9 章"字体",关于在 PM 应用程序中如何应用二者,请分别参照第 9 章和第 6 章。

1.1.2.6 标记图元

标记图元是图形实体,可用于指示折线图上的峰与谷。GPI 具有一个基本的标记集,该标记集包括可见的标记符号,例如正方形、菱形、星、圆及十字叉形。

标记图元存放在 MARKERBUNDLE 数据结构中。标记图元属性包括标记符号、标记框以及前面提到的标记集合、颜色及混合模式。

有关标记图元的详细情况及在 PM 应用程序中如何使用它们请看第 4 章"标记图元"。

1.1.3 颜色和混合属性

颜色和混合是图元的两个属性。颜色属性描述一条线、一段弧、字符、面域和图象的颜色。混合属性则决定每一图元是怎样与一已有的图形相结合,同时,它还在其它事物中,当不同颜色的图元重叠时,影响最终的颜色。

一些图元具有前景和背景颜色属性。例如,字符图元的前景颜色属性决定了字符的颜色,其背景颜色属性决定该字符周围的颜色。有前景和背景颜色属性的图元同样具有前景和背景混合属性。

关于用 GPI 实现彩色的具体情况请参看第 7 章"颜色和混合属性"。

1.1.4 位图

位图是存在内存当中的图形图象。使用图符编辑程序可以建立位图,也可使用 GPI 函数对位图进行硬编码。位图可以储存在内存设备描述表中或磁盘文件中。PM 既支持单色又支持彩色位图。

应用程序可以用位图完成下列事情:

- 存储和显示扫描进的图象、图符及指针;

· 给面域图元和路径建立填充模式；

· 使为多媒体而作的图画生动。

关于位图、图符和指针的详细情况，以及如何让 PM 应用程序的用户使用它们，请参看第 8 章“位图”。

1.1.5 元文件

元文件是存放描述图形的低层图形命令的文件。

OS/2 操作系统可以长期保留图形以使这些图形可以留给 PM 应用程序，甚至可以支持混合实体文档目录结构（MO:DCA）相互交换标准的非 PM 应用程序所使用。要在 PM 和非 PM 应用程序之间进行交换的图形数据是以元文件形式传到这些应用程序中。

关于元文件及它们在 PM 和非 PM 应用程序中使用的完整的详细情况，请参阅第 15 章“元文件”。

1.1.6 路径

路径是一实体，该实体由在 GpiBeginPath 与 GpiEndPath 之间的一系列图形图元和属性函数所定义。路径可用来定义填充域、复杂剪取形状，轮廓线，还可用来画几何（宽的）线。实际上，路径是画具有几何（可放缩的）宽度的直线，裁剪圆、椭圆或其它非矩形形状图形的仅有的手段，也是在轮廓字模中生成文本的有效方法。

要学会怎样用几种不同的绘图技术来定义和使用路径，请阅读第 10 章“路径”。

1.1.7 区域

区域是在应用程序设备空间中的矩形图形实体，该设备空间能用来裁剪输出图形，画一个或多个独立的已填充的矩形，或画包括相交矩形的已填充的多边形。区域也可用作重新填充一个窗口中的指定用户面域或一幅画中的部分实体。

区域是与设备相关联的并在设备坐标中定义。每个区域均为一个特定设备而建立，该设备是与显示空间相关联的。系统把区域当做设备描述表的一部分。

有关在 PM 应用程序中如何使用区域的详细情况，请阅读第 11 章“区域”。

1.1.8 保留的图形

在 OS/2 操作系统中有两类图形输出：保留和非保留。保留的图形是一显示空间模式，在此模式下，整个图形都存起来了，以供以后的显示和编辑。

应用程序可以用图形函数绘制非保留的图形输出。该输出图形在输出设备上（如窗口）立即显示。如果该图形的一部分被删除或者需重复，则应用程序必须如数地使用相同的函数；这是非保留图形的主要缺点。

保留的图形则提供了许多的优点，如方便、灵活、编辑能力强、更好和更有用的图形、及更好的组织结构。保留图形的缺点即为存储数据需使用额外的内存。

PM应用程序以段为单位存储图形。图形段是分组和存储图形图元和它们的属性的一种主要手段。虽然各图形段之间通常以某种方式相关联，但是它们并不一定非

得如此。

就有关如何建立和重新显示已保留图形的详细情况，请看第 12 章"建立和绘制保留图形"。关于如何编辑保存的图形及图形段的资料，请看第 13 章"编辑保留图形与图形段"。

1.1.9 关联

关联是确定一幅图形中的哪些图元或图元组应在显示中哪一位置的过程。关联大多用在图形应用程序中；例如，以识别由用户选择的图元或图元组。但是，应该注意到，关联也能用于非图形应用程序中；例如，当应用程序在计算器的运算模式下并允许用户选择数字进行数学运算的时候。

有关关联过程的低层的信息，请参阅第 14 章"相关检测"。

1.1.10 剪取和边界确定

剪取是一个过程。通过该过程，在所定义的剪取边界以外的图形都被丢失（剪取），仅显示在剪取边界之内（剪取区域）的那部分。

剪取边界通常由应用程序所规定。例如，当应用程序的图形输出被剪取以能适合于客户区域（用户工作区域）或者某硬拷贝设备的设备输出区域时，PM 的接口会自动地进行这一剪取。

边界确定是计算在屏幕上要包含一图形实体的最小矩形大小的运算，其目的在于当该图形实体移动或修改时，仅需修改屏幕上受影响的那部分。

对于 PM 应用程序的剪取和边界确定的综合资料，请看第 16 章"剪取和边界确定"。

1.1.11 坐标空间及变换运算与函数

变换使应用程序能够控制输出图形在任一输出设备上的大小和方向。GPI 的变换函数为图形和文本提供了放缩、变换、旋转和剪取功能。几乎所有这些 GPI 函数都在称为世界坐标空间的概念面域中绘制其图形输出。

下表介绍了 GPI 变换中用到的五种坐标空间：

表 1-1 坐标空间类型

名　称	说　明
World coordinate space	应用程序规定其用于画图的坐标空间。
Model space	概念空间，模式变换矩阵的结果在此空间中画图。
Default page coordinate space	表示一在页坐标定义的不能进行放缩和滚动的图的空间。
Page coordinate space	概念空间，缺省视图变换矩阵的结果在此空间中画图。
Device coordinate space	设备的坐标空间。

有关 GPI 图形变换的所有变换和函数的信息，请看第 17 章"坐标空间和变换"。

1.1.12 打印作业的提交与管理

打印是产生硬拷贝输出的过程。应用程序可以用缓冲区来提交作业，而该缓冲区将把提交的作业送到适当的打印机去打印输出。同样，应用程序也可以把作业直接送到打印机输出。直接提交会中止所有向该应用程序输入数据的用户直到打印结束为止。

若选择使用缓冲区，则当该作业排在缓冲区队列中时，还照样能操纵该作业。最典型的操纵就是增加要打印作业的份数或者将此作业完全删除。

有关如何打印作业及当作业在队列中时该怎样操作的更详细的情况，请阅读第 18 章“打印作业的提交和管理”。

1.1.13 程序执行

图形编程受所支持的操作系统的影响。OS/2 2.0 编程指南第 1 卷介绍了操作系统运行的概念，而以下则是 GPI 编程所涉及的一些概念：

- “节”是具有其自身的虚拟控制台的一个或多个过程。虚拟控制台就是一虚拟屏幕——或者是以字符为基础的整个屏幕或 PM 窗口——及键盘和鼠标输入用的缓冲区。
- “过程”是内存中应用程序的代码、数据及其它资源——如文件句柄、信号量、管道、队列等。操作系统把每一个它所加载的应用程序都当作过程。
- “线索”是一可分配的执行单元，它包括一系列的指令、相关的 CPU 寄存器值及堆栈。每个过程至少有一个线索并似在同一时刻运行多个线索。当操作系统把控制权交给过程中的线索时，相应的应用程序开始运行。线索是执行排队的基本单位。

图形输出慢，故最好是以异步线索模式完成很长的画图操作。这样，主线索可自由地处理其消息队列且仍然能对用户输入进行积极响应。用户的输入只到主线索，因此，若想达到对用户请求的快速响应，保持主线索与画图线索之间的良好通讯是很重要的。

例如，若用户要修改或删除图形，主线索必须能够中断异步线索的处理而去传递修改指令。主线索以 SDW_ON 值调用 GpiSetStopDraw 函数，这就挂起所有在 GpiSetStopDraw 中提到的图形显示空间的画图命令，并置为停止画图的状态。

在同一过程中及在不同的过程中往其它的显示空间绘制图形不受此停止画图状态的影响。但是，当停止画图状态有效时，从其它线索往所提到的显示空间画图是不可能的。当调用此函数时，所有往该同一过程中的其它线索处理的显示空间发出的画图操作都将终止并予以警告。当主线索收到画图线索的通知时，它可以用 SDW_OFF 值调用 GpiSetStopDraw 来清除该挂起。

1.2 小　结

以下是一些 GPI 的常用函数的小结。

表 1-2 GPI 函数

属性函数	
GpiMove	把当前位置移到指定点。
GpiQueryAttrs	返回指定图元类型的当前属性。
GpiQueryCharSet	返回由 GpiSetCharSet 设置的字符集局部标识符(ICID)。
GpiQueryCurrentPosition	返回当前位置值。
GpiQueryLineType	返回由 GpiSetLineType 设置的当前装饰线类型属性。
GpiQueryLineWidth	返回由 GpiSetLineWidth 所设置的装饰线宽度属性的当前值。
GpiQueryMix	返回由 GpiSetMix 设置的字符前景颜色混合模式的当前值。
GpiQueryPatternSet	返回由 GpiSetPatternSet 所置的模式集标识符的当前值。
GpiSetArcParams	置当前弧参数。
GpiSetAttrs	为指定的图元类型设置属性。
GpiSetCharBox	把当前字符框属性置为指定值。
GpiSetCurrentPosition	把当前位置置为指定点。
GpiSetLineType	置当前装饰线类型属性。
GpiSetLineWidth	置当前装饰线宽度属性。
GpiSetMix	为每一单独的图元类型置当前背景混合属性。
GpiSetPattern	置模式符号属性的当前值。
位图函数	
GpiCreateBitmap	建立位图并返回该位图句柄。
GpiDeleteBitmap	删除位图。
GpiQueryBitmapBits	把数据从位图传送到应用程序的存贮区。
GpiQueryBitmapHandle	返回由指定的局部标识符当前所标识的位图句柄。
GpiQueryBitmapParameters	返回由位图句柄所标识的位图信息。
GpiQueryDeviceBitmapFormats	返回由设备驱动程序内部支持的位图格式。
GpiSetBitmap	在内存设备描述表中置位图为当前所选位图。
GpiSetBitmapid	用局部标识符标识位图，使得该位图能用做包含单个成员的模式集。
设备函数	
DevCloseDC	关闭设备描述表。
DevEscape	使应用程序能够访问设备的资源，这些资源无法通过 API 得到。
DevOpenDC	建立设备描述表。

DevPostDeviceModes	返回并可根据需要设置作业优先权。
DevQueryCaps	查询设备特性。
DevQueryDeviceNames	查询显示驱动程序以返回它所支持的设备的名称、说明及数据类型。
DevQueryHardcopyCaps	查询设备的硬拷贝能力。
画图函数	
GpiBeginArea	开始建立一面域。
GpiBeginPath	指定路径的开始。
GpiBitBlt	复制一矩形位图图像的数据。
GpiCharString	从当前位置开始画出一字符串。
GpiEndArea	结束有阴影面域的构造。
GpiEndPath	结束由 GpiBeginPath 开始的路径说明。
GpiErase	把与指定显示空间相连的设备描述表的输出显示清除为基色。
GpiIntersectClipRectangle	把新的裁剪区域置为当前裁剪区域与指定矩形块的相交部分。
GpiLine	从当前点到指定点画一直线。
GpiPaintRegion	用当前模式属性把一区域颜色绘进显示空间。
GpiPolyLine	在当前点与所指定的各点之间画直线。
GpiPolySpline	建立一系列贝塞尔样条。
GpiQueryCharStringPos	把一字符串看成它正用 GpiCharStringPos 并以当前字符属性在绘制，然后返回字符串中每个字符要绘制的位置。
GpiQueryClipBox	返回能包括所有要裁剪域的相交部分的最小矩形的大小。
GpiQueryPel	返回在完全坐标系中指定位置的象素颜色。
GpiRectVisible	确定一矩形是否完全落在与指定显示空间相连的设备裁剪区域内。
GpiRestorePS	把显示空间的状态恢复到某一地方，该地方在相应的 GpiSavePS 调用时就存在。
GpiSavePS	把显示空间的信息存到 LIFO 堆栈中。
GpiSetClipRegion	通过指定的显示空间确定绘画时用于裁剪的区域。
GpiSetPel	使用当前(线)颜色及混合，设置在完全坐标系中指定位置的象素。
GpiWCBitBlt	复制一矩形域的位图图像数据。
字模函数	
GpiCreateLogFont	提供字模的逻辑定义。
GpiDeleteSetId	删除一逻辑字模或位图标记。

GpiLoadFonts	从指定源文件中装入一个或多个字模。
GpiQueryFontMetrics	返回为当前所选的逻辑字模提供详细的字模尺寸数据的记录。
GpiQueryFonts	返回一条记录，该记录提供与指定的 pszFacename 相匹配的字模的详细信息。
GpiQueryNumberSetIds	参照字模或位图，返回当前正使用的局部标识符的数目。
GpiQueryWidthTable	返回由字符集属性值所标识的逻辑字模的宽度表信息。
GpiSetCP	置缺省的图形码页。
GpiUnloadFonts	从内存中清除由 GpiLoadFonts 从源文件装入内存的所有字模。
显示空间函数	
GpiCreatePS	建立显示空间。
GpiDestroyPS	删除显示空间。
GpiQueryDevice	返回目前与设备描述表相关连的句柄。
GpiResetPS	重置显示空间。
GpiSetPS	置显示空间的大小、单位和格式。
区域函数	
GpiCombineRegion	合并两个区域。
GpiCreateRegion	用一系列的矩形为特殊设备类建立一区域。
GpiQueryRegionBox	返回能包含某区域的最小矩形的大小。
GpiSetRegion	把一区域变成一系列矩形的逻辑与(OR)。
变换函数	
GpiConvert	把一坐标对从一个坐标空间变换到另一个。
GpiQueryModelTransformMatrix	返回当前模式变换。
GpiSetModelTransformMatrix	为其后的图元置模式变换矩阵。
GpiSetPageViewport	置设备空间中的页观察口。

第 2 章 显示空间与设备描述表

显示空间是由操作系统管理的数据结构，与图形输出有关的信息都存在其中。这些信息既与随后的图画（如颜色或线型）有关又与显示空间的资源（如颜色表或字形）有关。

因为许多 GPI 函数必须在显示空间中运行，所以，在图形应用程序中，第一任务就是定义显示空间。应用程序对每一当前使用的输出设备都要求显示空间，包括在显示屏上的每个独立的窗口。因为每一可能的输出设备的要求都不相同，故几乎在所有情况下，显示空间都要置成设备独立。

> 注：在某些情况下，可以把显示空间从一设备脱离而与另一设备相连，通过此法而达到共享显示空间。

为便于 PM 编程接口的设备独立，所有与设备相关的专门信息都存放在设备描述表中。设备描述表是一数据结构，它辨识某类输出设备中的实例，并含有所有与设备相关的信息，如设备的逻辑名及显示驱动程序名。所有打算使用的输出设备的信息必须在设备描述表中描述。例如，若某应用程序使用了多个窗口，则每个窗口必须有它自己的窗口设备描述表。

所以，为把图形数据送到输出设备上，应用设备描述表必须与显示空间相连。关于如何建立这些联系的详细情况，参阅“把显示空间与设备描述表相关联”一节。

本章描述显示空间及设备描述表。本章内容包含下列主题：

- 图形图元
- 坐标空间
- 变换
- 图段

2.1 关于显示空间

有二类显示空间：

- 标准微型
- 高速缓存化的微型

· 普通型

2.1.1 微显示空间

如果应用程序为每一实际的输出设备建立一单独的显示空间且如果该应用程序的输出仅为画图时，则需要一微显示空间。

虽然根据应用程序使用的图形函数的不同，运行时内存需求的变化很大，但是典型的微显示空间图形应用程序通常使用 315KB 的 GPI，发生器和显示内存资源（代码和数据）。微显示空间不能与不同的设备再相连。

2.1.1.1 标准微显示空间

假如显示空间没有处于保留方式或保留与绘画方式，则标准微显示空间可用作在任何类型的输出设备上画图。

两个不同的函数可用作访问标准微显示空间。表 2-1 列出了这两个函数以及它们的用法。

表 2-1 建立标准微显示空间的函数

函数名称	用 法	关闭函数
GpiCreatePS	以输句柄、设备描述表句柄以及显示空间大小作为输入参数。建立普通显示空间和微显示空间。	GpiDestroyPS 删除一显示空间并释放该显示空间所占用的所有资源。
WinGetScreenPS	显示空间代表整个显示屏。警告：当使用这一函数作为图形输出可能覆盖单个的窗口时会提出警告。	WinReleasePS

2.1.1.2 高速缓存化微显示空间

窗口管理程序为显示屏上的窗口保留一显示空间的高速缓存。该高速缓存区是为使用多窗口的应用程序提供的，在这些应用程序中，为了输出准备的中间结果每个窗口都需要一临时的显示空间——设备描述表对。这些显示空间属于系统而非应用程序，且都是临时分配的。

高速缓存化的微显示空间由窗口系统而不是由 GPI 提供；它们的使用是与其它窗口活动同步的。例如，你无须把一高速缓存化的微显示空间与显示屏相连；因为窗口管理程序会完成这些工作。

因为与普通显示空间及标准微显示空间不同，高速缓存化的微显示空间不永久地分配给一个应用程序，所以它们能提供最佳的系统性能。PM 编程接口是模式接口，但是因为所有的属性都必须不断地初始化，所以高速缓存化的微显示空间使用起来有些繁锁。

高速缓存化的微显示空间可用作仅把输出送往显示设备上的窗口中。有三个不同的函数可以用来访问高速缓存化的微显示空间，每个函数均有其独到之处。这些函数列在表

2-2 中。

表 2-2 获得高速缓存化的微显示空间的函数

函数名	用 法	关闭函数
WinBeginPaint	为高速缓存化的显示空间接收-NULL 显示空间句柄。用该函数建立的显示空间已经预先同窗口设备描述表连接好了，这样使得该函数为最容易使用的函数。	WinEndPaint 自动释放该显示空间而不管它是什么类型。
WinGetScreenPS	显示空间代表整个显示屏。 警告：当使用该函数作为图形输出可能覆盖单个的窗口时会提出警告	WinReleasePS
WinGetPS	显示空间能代表整个工作台，或任何其它窗口。显示空间可用以处理任何消息，但当消息处理完成时，一定要返回到高速缓存区。	WinReleasePS

一般而言，当消息之间无须记住显示空间的信息时，可用高速缓存化的微显示空间来处理单个的绘图消息。在处理这条消息的过程中必须获取并释放该显示空间。一旦该显示空间释放回高速缓存区，存贮在高速缓存化的微显示空间中的所有应用程序的信息都将丢失。

必须给 WinBeginPaint 与 WinGetPS 提供有关输入的窗口句柄。由此而来的显示空间为此窗口专用，不能重新连接。

高速缓存化的微显示空间总是：

- 以象素来定义——虽然可用 GpiSetPS 改变其单位；
- 格式化为 GPIF_LONG；
- 由系统给一适当的大小。

当不需再用高速缓存化的微显示空间时，不一定要把它同窗口设备描述表脱离，因为 WinReleasePS 或 WinEndPaint 会完成这一脱离。这就使得该高速缓存化的微显示空间可被其它的窗口使用。显示空间自身无法删除。

高速缓存化的微显示空间是顺序使用的。你下次需要一高速缓存化的显示空间时，使用适当的函数访问一新的。每当你得到一高速缓存化的微显示空间时，图形属性都被置成它们的缺省值。

2.1.2 普通显示空间

如果要求你的应用程序使用相同的显示空间把输出送到多个设备上（例如显示屏和打印机），或者如果应用程序使用段和保留绘画函数来画复杂的图形时，则必须使用普通显示空间。

仅有一个函数用于建立普通显示空间：

表 2-3　建立普通显示空间的函数

函数名	用　法	关闭函数
GpiCreatePS	接收锚句柄，设备描述表句柄，显示空间大小作为参数。既建立普通显示空间又建立微显示空间。	GpiDestroyPS 删除显示空间并释放它。

使用普通显示空间，或执行元文件比使用微显示空间要化费更多的内存运行时间。普通显示空间比微显示空间多需 114KB 空间。

2.1.3　模块图形系统

图形图元，如直线或曲线，产生送往显示空间的图形输出。这些图元如何显示部分地取决于显示空间的模式。

在模块系统中，在发出绘制图元的指令之前，一些修改图形图元的信息就已经建立了。例如，图 2-1 就是从(4,3)到(9,7)画的红实线的例子。用 PM 的模块接口，需要向 GpiLine 函数提供的唯一值就是终点(9,7)。

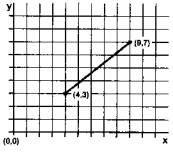

图 2-1　用 GpiLine 函数画的直线

该直线自动地用当前颜色，红色，以当前线型，实线，从当前位置(4,3)开始画。有单独地用于指定当前颜色、线型和位置的函数。

在非模块图形系统中，当发出一条指令时，要给出与该指令有关的信息。例如，发出一条画线指令，在此指令中需指明该线为绿色、用虚线，从 a 点画到 b 点。这些值或属性与一特定的画图指令连在一起，不影响其它指令。若要从 b 点到 c 点，则需重新指定“绿色”、“虚线”、“点 b”，及“点 c”。

对 PM 编程接口，图形图元的缺省值(或者在缺省值修改后的当前值)与当前位置一起存在显示空间中。首先，要完成第 1 步即要画直线，看来需要很多函数，方式图形系统实际上存贮有许多资源。在画完第一条线之后，如果还要画第二条或第三条线，则可能线的属性不变，即是，显示空间的方式仍相同。因此，在非方式图形系统中，如果所希望的输出要求 1 或 2 个以上的图形函数的话，属性的经常重复实际上要用更多的资源。

可以定义图形并存贮其定义而不必把它们送到屏幕或打印机上。在这种情况下，属性流的概念变成相关了。例如，屏幕上线的颜色就是你定义该直线时的颜色。其颜色并不总

是画该直线时的系统当前颜色。

2.1.4 当前位置

当前位置是完全坐标空间位置，下一个画图请求便从此点开始；它是存在显示空间中数据的一部分。当建立或获取显示空间时，当前位置设成为原点(0,0)。当显示空间与不同的设备描述表相连时，当前位置也重置为(0,0)。

大多数画图请求都改变当前位置。例如，若从原点到(3,5)画条线，则画完之后的当前位置就变为(3,5)。用 GpiSetCurrentPosition 或 GpiMove 函数能直接改变当前位置。若不想在显示空间中从原点(0,0)开始画图，请先调用 GpiSetCurrentPosition 函数。

2.1.5 图元属性

图形图元以及画这些图元的函数在第 3 章到第 9 章中讨论。图元函数使图元的属性被置定，以为了所有该图元以后的调用。这些属性存贮在称为 bundles 的数据结构中。在图元的绘制执行之前，有些函数可以设置图元的属性。GpiSetAttrs 函数不仅可以指定图元，属性将为该图元而设，而且可以指定用于特定属性的图元集。GpiQueryAttrs 为指定的图元返回指定属性的当前值。

另外两个函数，GpiSetAttrMode 和 GpiQueryAttrMode，能够定义或查询属性方式设置，AM_PRESERVE 和 AM_NOPRESERVE。AM_PRESERVE 设置保留有某属性所有改变之前的值。例如，若要画 6 条虚线，1 条实线，然后再 6 条虚线，则原先的值很容易恢复。AM_NOPRESERVE 设置则在属性值一旦改变即放弃该属性的原值。属性值是显示空间的资源。

2.2 使用显示空间

显示空间函数可以用来：

- 建立普通的、微或高速缓存化的微显示空间；
- 删除、保存和恢复显示空间；
- 定义或判定显示空间属性。

2.2.1 获取和建立显示空间

关于如何获取和建立显示空间的详细资料和样本源代码，请参考 OS/2 2.0 编程指南第Ⅱ卷。

2.2.2 删除显示空间

显示空间占用相当数量的内存；故应用程序不再需要显示空间时应该删除它。可以用 GpiDestroyPS 函数来删除普通或微显示空间。

当不再使用靠 WinGetPS 函数访问的高速缓存化的微显示空间时，用 WinReleasePS 函数释放它。不必去删除用 WinBeginPaint 访问的显示空间。调用函数 WinEndPaint，则

PM 会完成这些操作。

2.2.3 保存和恢复显示空间

你可以保存一些显示空间的属性和资源，修改它的域，在修改后的显示空间中画图，然后用 GpiSavePS 和 GpiRestorePS 函数恢复它。当你调用 GpiSavePS 时，图形工具从当前显示空间中将下列项目复制到特定的堆栈中去：

- 图元属性
- 变换矩阵
- 观察范围
- 裁剪路径
- 裁剪区域
- 当前位置
- 已装入的逻辑颜色表
- 已装入的逻辑字模

可以根据需要把显示空间的内容多次压入堆栈。GpiRestorePS 把显示空间的内容弹出堆栈。

2.2.4 显示空间小结

表 2-4 总结了每类显示空间的特点和局限。

表 2-4 显示空间的特点和限制

特点/限制	普通型	标准微型	高速缓存化微型
所支持的设备类型	任意设备	任意设备	仅限于视频显示窗口
支持设备数	多个	一个	一个
关联	根据需要进行关联与脱离	在建立时关联；不能脱离	N/A
保存图形	是	否	否
可提供的 GPI 函数	全部	除段函数以外的全部	除段函数以外的全部
内存	最多的内存使用	中等内存的使用	最快的分配

2.3 关于设备描述表

设备描述表通过把与设备无关的显示空间的信息转换成与设备相关的信息而把显示空间与设备连接在一起。该转换在显示驱动程序(presentationdriver)中完成，显示驱动程序是对应用程序透明的低层程序。显示驱动程序是一段设备驱动程序代码，此代码把应用程序的命令转换成为对每一输出设备的特定输出命令。

在建立设备描述表之后，在图画(或图元)打印或显示之前，设备描述表必须与显示空间相连。当显示空间与设备描述表连好之后，送到显示空间的所有输出都自动送往设备描

述表并在物理设备上显示。例如，在显示窗口画一图画之前，该图的图形显示空间必须与窗口的设备描述表相连。

设备描述表还提供应用程序访问象屏幕大小或打印机这样的重要设备信息的能力。

和显示空间一样，也有三类设备描述表：

- 高速缓存化型
- 窗口型
- 普通型

2.3.1 高速缓存化的设备描述表

高速缓存化的设备描述表是空间，该空间在从高速缓存区获得时便已与高速缓存化的显示空间相连在一起了。

2.3.2 窗口设备描述表

窗口设备描述表是一特殊类，特殊在于它们是仅有的用 WinOpenWindowDC 获取的设备描述表。正如它们的名字所示，它们所代表的输出设备是显示窗口。WinOpenWindowDC 以窗口句柄为输入，并返回窗口设备描述表句柄。这样，窗口设备描述表句柄可以与标准微显示空间或普通显示空间相连了。

在窗口设备描述表相关连的窗口删除之后，该描述表也就自动关闭了。

2.3.3 普通设备描述表

普通设备描述表（也称为标准或非高速缓存化的设备描述表）把显示空间与任一非窗口输出设备相连。用 DevOpenDC 函数获取列在表 2-5 中 6 个普通设备描述表中的一个设备描述表的句柄。在应用程序结束之前必须用 DevCloseDC 关闭 6 个普通设备描述表中的每一个。

表 2-5 普通设备描述表

DC 类型	目 的	用 途
Queued	把显示空间与由打印队列中送缓冲打印作业的多个应用程序共享的打印机或绘图仪相连接。	应用程序利用排队的设备描述表以从应用程序脱开打印控制。
Direct	直接绕过缓冲区和打印队列，把显示空间与打印机与绘图仪相连。缓冲区用直接设备描述表来处理那些从打印队列中移出的作业。	除非（为保密起见）应用程序要避免使用队列或直接送到专用机器上，否则应用程序通常不用直接设备描述表。
Information	把显示空间同打印机或绘图仪相连，直接使设备的信息可以被查询，但在设备上不产生输出。	应用程序能使用具有低存贮的信息设备描述表而不用直接设备描述表，该表能提供相同的信息。