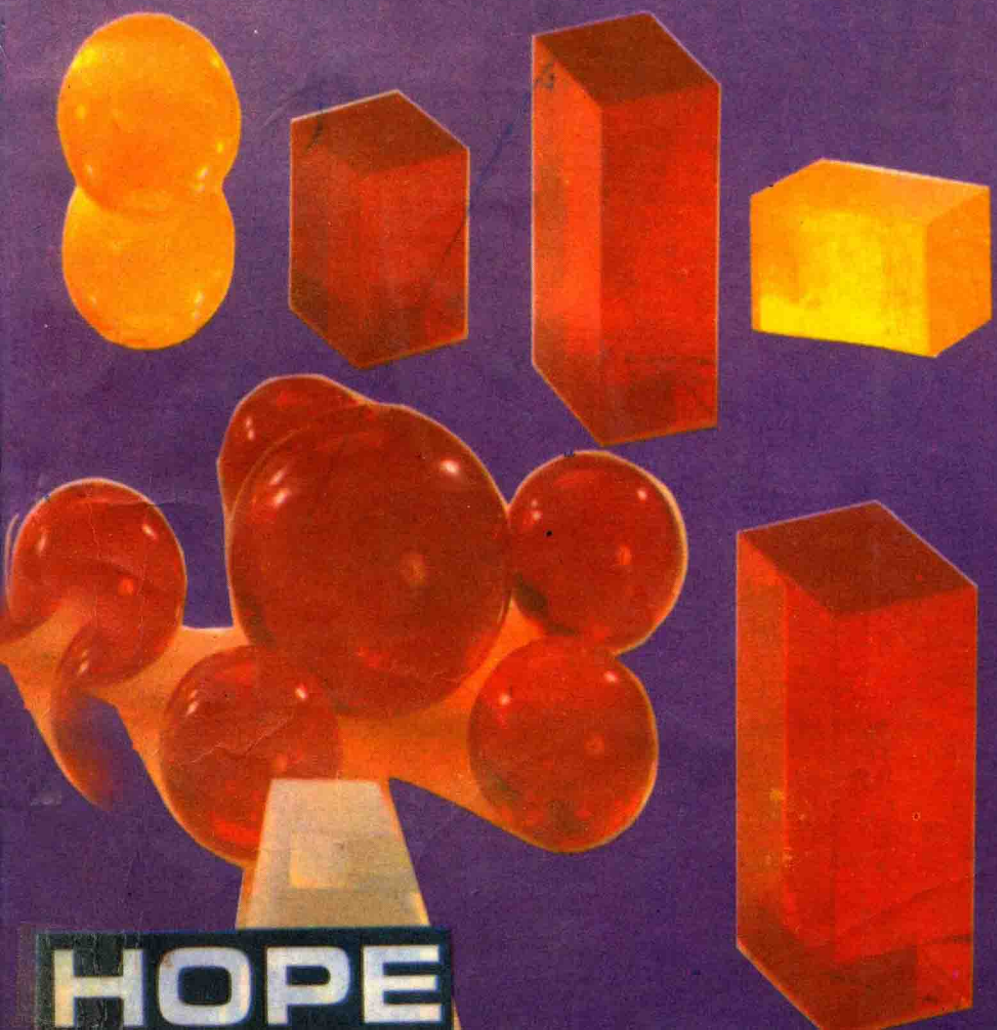
OBJECT-ORIENTED PROGRAMMING

Turbo Pascal 5.5～6.0版
图形程序设计方法和技巧



北京希望电脑公司

HOPE

Turbo Pascal 5.5 ~ 6.0

图形程序设计方法和技巧

田青　步越　吴子平　　编

北京希望电脑公司

一九九一年八月

目　录

第一章　视频图形介绍

在计算机产业中，盛行着这样一个神话：pascal 不是面向图形的语言，是不能胜任图形应用的。也许，在过去的计算机年代里，确如此话所言，因为计算机曾一度不是面向图形的（除早期的 Apple 系统：它面向图形，但不面向应用）。

但是，昨日的神话已非今日之现实。视频系统已从原先仅有模糊图形功能的 CGA 系统大大地扩展。早期的 2~MHZ、16K 系统，对图形应用讲其速度、存储器都不够，如今事实上都已过时，被具有兆字节、更大存储器的 20/30MHZ 系统所代替。

同时，图形接口和应用程序，象 Microsoft Windows、os/2 和 Ventura Publisher 都很流行，很普遍。

最为重要的是，Turbo Pascal 出现了。它不仅是面向图形的编译程序，而且，使用 Borland 图形接口（Borland Graphics Interface）时，就如同一个图形巨星。它提供了过程和函数以支持图形应用和使图形应用走向实用所必需的速度，这正是我们需要的。

图形学不再只是用于以幻想的装饰装点传统的程序，或是为拱廊游戏提供色彩和图象。相反，图形学已不仅是成为一个新标准，而且，将依据这一标准，对许多（即使不是大多数）的未来应用作出判决。这绝非一时的潮流。甚至便携式计算机，虽然目前大多仍缺乏色彩功能，也都迫不及待地匹配或模仿高分辨率图形显示。例如，我的最新系统是采用背景照亮，超扭曲液晶屏幕的 VGA 视频，其色彩模仿为 16 级灰度，总重 14 磅，清晰度极好，随处使用，胜于同类相当的产品。事实上，这本书里的许多排版和图表（两者如没有很好的图形功能是做不到的）就是在我的膝上便携机上完成的（请想象一下）。

如果甚至便携式计算机都能支持强大的图形，难道这不意味着图形的实现？

1.1　为什么使用图形显示？

尽管一个程序的总体外观，特别是在竞争激烈的商业环境中，是一个重要因素，图形显示的优越性却远不只是装点一下门面。

今天，Graphic User Interface 或 GUI，一般用以泛指任一应用或系统，征用图形屏幕元素和鼠标选择，而非仅依靠键盘输入控制。总的来说，GUI 元素被设计成易于辨识，无需解释或说明文件，比人们熟悉的 C:>提示符提供更多面向用户的对话环境。在有些情况下，例如积极面向图符的 Macintosh 机，在这一点走向极端，许多人都觉得很烦琐。

抛开极端，一个平衡的图形和文本元素结合体允许用方便、快速理解的形式提供信息。例如，条状的光标可显示长表或页作用域之内的位置，也提供改变显示位置的鼠标控制和鼠标操作按钮（改变状态以显示选择）功能，这就是极好的图形元素，操作中无需解释。

在同样情形下，商用图形比一列列数字更易于比较、理解。在许多数学应用中，一个好的三维图形揭示了整个理解的世界，而这些用原始数据或公式是做不到的。至于排版软件，几乎没有图形的替代物——参见流行的 Word Perfect 5.0——它提供了标准的文本显示和图形排版的页输出（包括图象）。

另一方面，无论完成什么工作都得把精制的折叠线和剪裁线图象拉到屏幕周围，这可

能会很恼人。显然这需要平衡一下。

但是，在担心使用什么之前，首先需要的是能产生图形显示。

选择图形显示器

现今丰富的视频硬件（如今包括至少 10 种标准视频适配器）给编程者带来了其自身的问题。每一种设计的视频硬件支持不同系列功能，也可能使用不同的存储器地址，可以或不可以支持多图形页（多文本页），可能支持的视频模式为从 320*200 到 1074*768 象素显示以及从单色到 256 色显示。

当然，每一个图形程序员遇到的初始问题仅是确定机器及安装的是何种视频适配器。一种方案常常是询问终端使用者，允许他们选择将要使用的视频模式。这种选择很难，因为甚至专业程序员也不总是能确信他正在使用的是什么硬件，而一般终端使用者也许干脆就不知道他是否有图形适配器，更不用说是何种型号了。另一种选择是由软件查询硬件以识别当前的配置。

假如已有了一些识别硬件的标准，这就很简单了。但是，对于编程员来说，虽然丰富的视频适配器出现了，遗憾的是，却没有任何正式的识别规定。在早期的书《IBM 用户接口编程》中，我们讨论了几种测试 CGA/EGA 视频适配器存在与否的方法，以及正确识别硬件中的一些内在问题。

好在已出版的 Turbo Pascal Version 4.0（和 Turbo C 1.5）已经解决了许多识别、支持和使用现今用得到的各种视频适配器中的未决问题。从 Borland 的过去表现来看，似乎有相当把握假定新的视频适配器将同样被支持。

但是，这种颇受欢迎的对不同视频容量给予的支持也产生了自身的混乱，即如何使用这些新工具，哪些是可能的，总起来，即怎么处理过剩的新能力。

这种混乱正好就是本书讨论的主题。

1.2 视频适配器类型

Turbo Pascal（4.0 版本或更新）和 Turbo C（1.5 版本或更新）都为所有主要的、现今在使用类型的视频卡提供完全支持。这次，我将只讨论和阐明 Turbo Pascal 中的图形部分。如果你正使用 Turbo C，并且基本上使用同样的函数名和过程名、以同样的方式有效地运行，那么就可提供相应的功能。

当今视频卡，从仅有文本功能的原始视频系统到超高分辨率的 IBM－8514（以排版和 CAD 软件应用而流行），还具有一系列分辨率介于两者之间的类型。高分辨率视频卡必须与具有相应象素分辨率的监视器相匹配，但是这是硬件的问题，对作为编程者的所有实用目标而言。你可以简单地假定当前的硬件与从 Turbo Pascal 的 Detect Graph 函数反馈的识别信息相符。任何匹配视频卡和监视器的不符，都是终端使用者的责任，而不应是编程者所关心的。

即使不是全部，也有大多数的多路同步和高分辨率的图形视频卡对部分实际监视器类型予以承认。

1.3 视频模式

每台 PC、XT 或 AT 都配置了某种类型视频适配卡。从非常简单的视频卡开始，有

Monochrome Display Adapter（MDA），支持仅用于文本的显示。如果是这样，你就不能编制或使用图形，除非对你的系统升级。

再高级一点儿的是流行的 Color Graphics Adapter（CGA）卡，而更高分辨率和更宽选择有 Hexcules Monochrome Graphics Adapters、Multi Color Graphics Array（MCGA），和 Enhanced Graphics Adapter（EGA）视频适配器。对更高级的图形功能，如为排版或 CAD 应用系统 AT&T 400-Line Graphic Adapter，Variable 或 Video Graphics Array（VGA）（有时称作 Multisync 视频），PC-3270，和 IBM-8514 视频适配器都提供了更高的象素分辨率。

在 Turbo Pascal 中，以 6 个图形接口（.BGI）单元（ATT，CGA，EGAVGA，HERC，IBM8514，和 PC3270）和 4 种图形字体（GOTH.CHR，LITT.CHR，SANS.CHR 和 TRIP.CHR）的形式提供了对所有这些类型图形的支持。当新的视频图形卡出现时，新的.BGI 单元可以被包括在内以便予以支持，但新的.CHR 字体也可由用户自己产生或从商业来源中购得。（参见第十七章，附录 D 和附录 E）

然而，图形支持单元并未在发行时包括在标准存储模式中——省略的特殊目的是在不需要图形时加速编译。

1.4 缺点

一旦程序编译完，.EXE 程序可以和运行所需的外部.BGI 和.CHR 文件分配在一起。这些文件一共大约需 60K 磁盘空间。当然，从存贮量角度来看，这不是过分的要求，但依靠外部文件来执行可能产生困难。

首先，调用 InitGraph 必须包含确定.BGI（和.CHR）模块位置的驱动和路径。如果没有指定路径，那就取当前目录。通常，查找信息由程序员提供，并且，如果所需文件没找到（这个问题可能源于很多因素，不管程序设计得多好），那么你的程序将中止！

其次，如果缺省的（当前的）路径被选择，所有外部文件都已具备，但程序调自其他目录或驱动器，这也将导致失败！

第三，不要指望终端使用者注意到外部文件的重要性。他们可能会喜欢你的程序，谨慎地保护好，可一旦空间成问题，就可能会删除一些必要的外部文件。这有时确实会发生，应该做好准备。另一方面，你可以把.BGI 和.CHR 文件直接连接到图形库中，这使你的.EXE 文件长度增加约 30K（参见第九章，Advanced Graphics Management Functions），并使驱动程序和字体成为.EXE 程序的一部分，而非外部文件。

另一种选择：如果连接这些文件产生太大的程序，不妨试试把外部文件设置为只读（Read Only）、系统（Systemr）或隐藏（Hidden）属性。

1.5 单元使用

Turbo Pascal 提供几种单元——为特殊应用提供扩展了的支持补充函数库——如今包括 DOS，GRAPH，GRAPH3，OVERLAY，PRINTER，SYSTEM 和 TURBO3 单元。用户单元也可用以产生新函数和新过程。用户单元和销售的单元使用一样方便。

使用单元——选择 GRAPH 单元为例，因为它是本书中所有程序的基础——自身简洁，而且除了在程序首部加上

```
uses GRAPH;
```

语句外，要做的事很少。

当需要几个单元里的函数时，在一行里可以调用多个单元，如：

```
uses CRT, DOS, GRAPH, TURTLE;
```

单元的调用顺序是无关紧要的，而且可以包含任意数目的单元。前一个例子也包含了一个用户单元，TURTLE，将在这本书里生成。

包含目录的选择

如果单元文件不在你的 Turbo Pascal 根目录里，Turbo 的集成调试环境保证获得指定 Turbo，EXE&TPU，Include，Unit 和 Object 文件的子目录路径（参见 Turbo Pascal 用户手册中“目录选择”部分）。

要指定单元目录途径可从菜单中选择 Options，Directory 和 Unit 各项，输入全路径以确定单元文件（.TPU）位置。有多个可能路径可以确定时，可采用分号（；）隔开，如下：

```
\TP\UNITS; \TP\EXE; \TP\OBJECT
```

这个例子指定了三个可能找到单无的子目录，即\TPUNITS 目录；\TP\EXE 目录——新编译的.TPU 文件会出现在这里，因为这个子目录是在 EXE 选择下指定的；以及\TP\OBJECT 目录.Pascal 根目录\TP 缺省地包含在里面。

1.6 注意：有什么？

图形程序的第一步是初始化合适的图形驱动程序。有关支持的图形视频卡、驱动程序和图形模式的说明，参见表 1-1。

表 1-1 Turbo Pascal 支持的视频模式

GRAPHICS DRIVER[1]	VALUE	GRAPHICS MODES	KEY VALUE[2]	COLUMN X ROW	PALETTE OR COLORS[3]	VIDEO PAGES
DETECT	0	requests InitGraph to execute autodetection				
CGA	1	CGAC0	0	320x200	C0	1
		CGAC1	1	320x200	C1	1
		CGAC2	2	320x200	C2	1
		CGAC3	3	320x200	C3	1
		CGAHI	4	640x200	2 colors	1
MCGA	2	MCGAC0	0	320x200	C0	1
		MCGAC1	1	320x200	C1	1
		MCGAC2	2	320x200	C2	1
		MCGAC3	3	320x200	C3	1
		MCGAMED	4	640x200	2 colors	1
		MCGAHI	5	640x48	2 colors	1
EGA	3	EGALO	0	640x200	16 colors	4
		EGAHI	1	640x350	16 colors	2
EGA64	4	EGA64LO	0	640x200	16 colors	1
		EGA64HI	1	640x350	4 colors	1
EGAMONO	5	EGAMONOHI	3	640x350	2 colors	1
IBM8514[5]	6	IBM8514LO	0	640x480	256 colors	1
		IBM8514HI	1	1024x768	256 colors	1

HERC	7	HERCMONOHI	0	720x348	2 colors	4
ATT400	8	ATT400C0	0	320x200	C0	1
		ATT400C1	1	320x200	C1	1
		ATT400C2	2	320x200	C2	1
		ATT400C3	3	320x200	C3	1
		ATT400MED	4	640x200	2 colors	1
		ATT400HI	5	640x200	2 colors	1
VGA	9	VGALO	0	640x200	16 colors	4
		VGAMED	1	640x350	16 colors	2
		VGAHI	2	640x480	16 colors	1
PC3270	10	PC3270HI	0	720x350	2 colors	1

注：1. Graphic Driver(图形驱动程序)和 GraphicMode(图形模式)名是由 GRAPHICS.H 定义的常数，即相应的数字和模式数值（见注 2）。

2. 模式设置为 InitGraph，DetectGraph 或 GetGraphMode 返回值。

3. C0…C3 代表 4 种预先定义好的调色板——见 SetPalette。

4. 64K 的 EGAMONO 卡，只能支持一个视频页，而 256K 则支持 2 个视频页。

5. 自动辨识不能正确地确认 IBM-8514 图形卡，而 InitGraph 或 DetectGraph 把 IBM-8514 卡确认为 IBM-8514 能正确模仿的 VGA 图形（IBM-8514LO 是与 VGAHI 等价的）。如使用高分辨率模式（IBM 8514HI 有 1024*768 个象素），在调用 InitGraph 之前，将 IBM-8514 值（在 GRAPHICS.H 中定义的数值 6）设定给图形驱动程序变量。不要将 DetectGraph 或是 DETECT 与 InitGraph 一起使用。也可参见 IBM-8514 和 SetRBGPalette 注释。

1. DetectGraph

通常，函数 DetectGraph 由 InitGraph 调用，但在确定当前图形驱动程序和图形模式时，也可被独立调用。

```
uses GRAPH;
var
   GraphDriver, GraphMode: integer;
begin
   GraphDriver := DETECT;
   DetectGraph( GraphDriver, GraphMode );
   ...
end;
```

如果出现问题，Graph Driver 返回出错代码；否则，GraphDriver 确定正确的驱动程序类型，GraphMode 返回该驱动器的最高有效视频模式。DetectGraph 不需要驱动程序路径（参见 InitGraph）。

DetectGraph 函数不能初始化任意一种图形设备。直接调用 DetectGraph，主要是因为随后将由 InitGraph 调用一个特定的图形驱动程序，或是选择一个缺省状态下 InitGraph 不会调用的图形模式。但使用 SetGraphMode 函数。初始化后，可以调用不同的模式。

返回值：GraphDriver 返回驱动程序类型或出错代码；GraphMode 返回最高有效视频模式。

可移植性：仅在 IBM PC 及兼容机上，相应函数存在于 Turbo C 中。

2. InitGraph

使用 Turbo Pascal 时，InitGraph 函数用以置初始图形参数值，装载适当的图形驱动程序，设置系统为所需的图形模式。

```
uses GRAPH;
const
   DriverPath = '';
var
   GraphDriver, GraphMode : integer;

begin
   GraphDriver := DETECT;
   InitGraph( GraphDriver, GraphMode,
              DriverPath );
   ...
end.
```

设置 GraphDriver 为 0（DETECT），指示 InitGraph 调用 DetectGraph（GraphDriver, GraphMode）以确定所安装的视频图形适配器的类型（和设置）。如果出错，GraphDriver 返回错误代码以指示错误类型，表 1-2 列出了图形错误代码。

表 1-2　图形初始化错误代码

代码	信息
−2	没有检测到图形卡
−3	没有发现图形驱动程序文件
−4	无效的驱动程序（或不被承认）
−5	没有足够内存加载图形驱动程序

DetectGraph 和 GraphResult 函数返回与表 1-2 所示相同的错误代码。

如没有错，内部错误代码就被设置为零，InitGraph 分配内存容量给适当的图形驱动程序，从磁盘中装载所需的.BGI 文件，并设置缺省的图形参数值。同样，GraphDriver 返回驱动器类型，而 GraphMode 返回模式的设置。

GraphDriver 和 GraphMode 也可通过使用适当的数字常量，或使用定义在 GRAPH.TPU 单元中已定义好的驱动程序和模式名来确定。在每一种情形下，Driver Path 显示.BGI 图形驱动程序所在的驱动器和路径。如果 Driver Path 为空，这些文件一定在缺省的目录里。如果它们在不同的目录里，那么完整的确定路径方式应显示如下：

Driver Path='\TP\BGI';

因为 InitGraph 设置的 Driver Path 也为 SetTextStyle 所用，用以寻找字体（.CHR）文件，所以.BGI 和.CHR 文件必须在同一个目录里。

返回值：Graph Driver 返回驱动器类型或错误代码；GraphMode 返回最高有效视频模式。

可移植性： 仅限 IBM PC 及兼容机，相应函数存在于 Turbo C 中。

3. GetDriverName

函数 GetDriverName 返回有当前图形驱动程序名的字符串。

例如： uses GRAPH;

OutTextXY（50，50，'Using Driver'+GetDriverName）；

GetDriverName 函数只能在 InitGraph 设置图形驱动程序和模式之后使用。详见 GetModeName。

可移植性： 仅限 IBM PC 及兼容机，相应函数存在于 Turbo C 中。

1.7 图形错误函数

如上所述，如果缺图形视频卡，或者没有找到图形驱动程序，或在初始化检查时出的一些其他错误，都 将由 DetedtGraph 或 InitGraph 返回错误代码。然而，其他情况下也可能出现图形错误，函数 GraphResult 和 GraphErrorMsg 就是用来测试和显示相应的错误结果和信息的。

1. GraphResult

函数 GraphResult 返回由报告出错的最后一次图形操作所设置的数字错误代码。这些错误代码从-15 到 0 的整型值。因为调用 GraphResult 时，出错状态被复位为零，所以返回值应存于局部变量中，然后检测之，以供进一步操作。

```
uses GRAPH;
var
   ErrorNumber;

ErrorNumber = GraphResult;
```

返回值： GraphResult 返回错误代码（－15 到 0），见表 113 的说明。

可移植性： 仅限 IBM PC 及兼容机，相应函数存在于 Turbo C 中。

表 1-3 图形错误信息

错误代码	图形错误常量	错误信息字符串
0	grOK	没有错误
－1	grNoInitGraph	没有安装.BGI图形（用InitGraph）
－2	grNotDetected	没有检测到图形硬件
－3	grFileNotFound	没找到设备驱动程序（.BGI文件）
－4	grInvalidDriver	无效的设备驱动程序
－5	grNoloadMem	没有足够内存加载驱动程序
－6	grNoScanMem	Scan填充内存不足

−7	grNoFloodMem	Flood填充内存不足
−8	grFontNotFound	没找到字体文件（.CHR）
−9	grNoFontMem	没有足够的内存加载字体文件
−10	grInvalidMode	驱动程序的无效图形模式
−11	grError	图形错误（普通错误）
−12	grIOerror	图形I/O错误
−13	grInvalid Font	无效的字体文件
−14	grInvalid RFontNum	无效的字体号
−15	grInvalid DeviceNum	无效的设备号

Graphics Errors 常量和错误信息在 GRAPH 单元定义。

2. GraphErrorMsg

GraphErrorMsg 函数返回一个指向相应错误信息字符串的指针。这些字符串由图形库（GRAPHICS.LIB）定义，但对于独立的错误信息过程，则能按所希望的显示出更为完全、更富信息量的错误信息。

```
uses GRAPH;
var
   ErrorNumber;

ErrorNumber = GraphResult;
writeln( GraphErrorMsg( ErrorNumber ) );
```

返回值：无

可移植性： 仅限于 IBM PC 及兼容机，相应的函数存在于 Turbo C 中。

1.8 其它图形模式函数

除了 EGAMONO、HERC 和 PC3270 视频驱动程序外，每种视频驱动程序支持两个或更多的视频模式，提供了多种象素分辨率或是各种色调。为处理模式查询、改变操作模式，Turbo Pascal 提供了多种函数：

1. GetGraphMode

函数 GetGraphMode 返回整型值，以显示 InitGraph 或 SetGraphMode 所设置的当前（工作）图形模式。

```
uses GRAPH;
var
   CurrentMode : integer;

CurrentMode = GetGraphMode;
```

返回值：当前图形模式。

可移植性： 仅限 IBM PC 及兼容机，相应的函数在 Turbo C 中也有。

2. GetModeRange

函数 GetModeRange 由一整型值调用，该整型值指明了图形驱动程序（可能是整型变量，也可能是 GRAPH 单元所定义的常量中的一个）；返回两个用以定义指定驱动程序的所支持的最小和最大有效模式的数值。

```
uses GRAPH;
var
   LoMode, HiMode : integer;
GetModeRange( GraphDriver, LoMode, HiMode );
```

如果 GraphDriver 传送的值无效，那么 LoMode 和 HiMode 返回值为－1。

返回值： 最小和最大有效模式；或错误代码－1。

可移植性： 仅限 IBM PC 及兼容，相应函数在 Turbo C 中也有。

3. GetMaxMode

函数 GetMaxMode 返回对当前加载的驱动程序的有效最高模式字，直接从驱动程序中得到返回值。函数 GetModeRange 仍被支持，但仅对 BGI（Borland 支持的）驱动程序有效。

```
uses GRAPH;
var
   MaxMode : integer;
MaxMode := GetMaxMode;
```

所有驱动程序支持模式零到 GetMaxModes，返回值是所能传送到 SetGraphMode 过程中的最大值。

可移植性： 仅限 IBM PC 及兼容机，相应函数在 Turbo C 中也有。

4. GetModeName

函数 GetModeName 由一模式值调用，返回当前图形驱动程序相应的模式名。

```
uses GRAPH;
var
   i : integer;
OutTextXY( 50, 50, 'Valid modes are: ' );
for i := 0 to GetMaxMode do
   OutTextXY( 60, 65 + 10 * i,
              GetModeName( i ) );
```

模式名嵌在每个图形驱动程序里，可用作菜单，以显示状态或报告系统容量。

可移植性：仅限 IBM PC 及兼容机，相应函数存在于 Turbo C 中。

5. GraphDefaults

函数 GraphDefaults 把所有图形设置复位为它们的缺省值（即由 InitGraph 初始设置和由 GRAPH 单元定义的值）。这包括把图形视口（图形窗口）复位为全屏幕；移动当前位置到（0，0）；复位缺省调色板、背景色及绘图色；复位缺省文件和图样类型；复位缺省

文本字体和排齐方式。

例：
```
uses GRAPH;
GraphDefaults;
```

返回值：无

可移植性：仅限 IBM PC 及兼容机，相应函数在 Turbo C 中也有。

6. SetGraphMode

图形模式必须由 InitGraph 事先初始化。函数 SetGraphMode 必须由对当前设备的驱动程序有效的图形模式调用（可使用 GetGraphMode 查找当前模式值或 GetModeRange 检查允许值）。一旦被调用，SetGraphMode 选择一个新的图形模式，清屏幕，把所有图形变量复位为其缺省值（见 GraphDefaults）。

```
uses GRAPH;
var
   ModeNumber : integer;
SetGraphMode( ModeNumber );
```

函数 SetGraphMode 也可以和函数 RestoreCrt 一起使用以实现文本与图形显示之间的相互切换。在使用这两个函数之前必须先调用 InitGraph。

```
uses GRAPH;
var
   CurrentMode : integer;
CurrentMode = GetGraphMode;
RestoreCrtMode;                    /* text mode   */
SetGraphMode( CurrentMode );     /* graphics mode*/
```

如果调用 SetGraphMode 的值对当前设备驱动程序无效，GraphResult 将返回值－10（grInvalid Mode）。

返回值：无，参见 GraphResult 的错误代码。

可移植性：仅限 IBM PC 及兼容机，相应函数在 Turbo C 中也有。

7. RestoreCrtMOde

函数 RestoreCrtMode 将系统视频复位到调用 InitGraph 时检测出的初始文本模式。这可以和 SetGraphMode 一起使用，以实现文本与图形显示之间的切换。

例：
```
uses GRAPH;
RestoreCrtMode;
```

返回值：无，参见 GraphResult 错误代码。

可移植性：仅限 IBM PC 及兼容机，相应函数在 Turbo C 中也有。

8. CloseGraph

函数 CloseGraph 将系统复位到由 InitGraph 初始检测的正常文本模式。也将释放驱动程序、字体和内部缓冲区所占用的图形系统的存储空间。

例：
```
uses GRAPH;
```

CloseGraph;

如果你希望对文本、图形作相互切换，可使用 RestoreCrtMode 和 SetGraphMode 函数。

返回值：无，参见 GraphResult 错误代码。

可移植性：仅限 IBM PC 及兼容机，相应的函数在 Turbo C 中。

FIRSTGRP.PAS

```
{====================================}
{            FIRSTGRP.PAS            }
{  demo for initializing graphics    }
{  mode in Turbo Pascal version 5.5  }
{====================================}

uses GRAPH, CRT;

type
   Str10 = string[10];   Str20 = string[20];

const
   DriverNames : array[ 0..10 ] of Str10 =
      ( 'Detect', 'CGA', 'MCGA', 'EGA', 'EGA64',
        'EGAMono', 'IBM8514', 'HercMono', 'ATT400',
        'VGA', 'PC3270' );
   Fonts : array[ 0..4 ] of Str10 =
      ( 'Default', 'Triplex', 'Small', 'SansSerif',
```