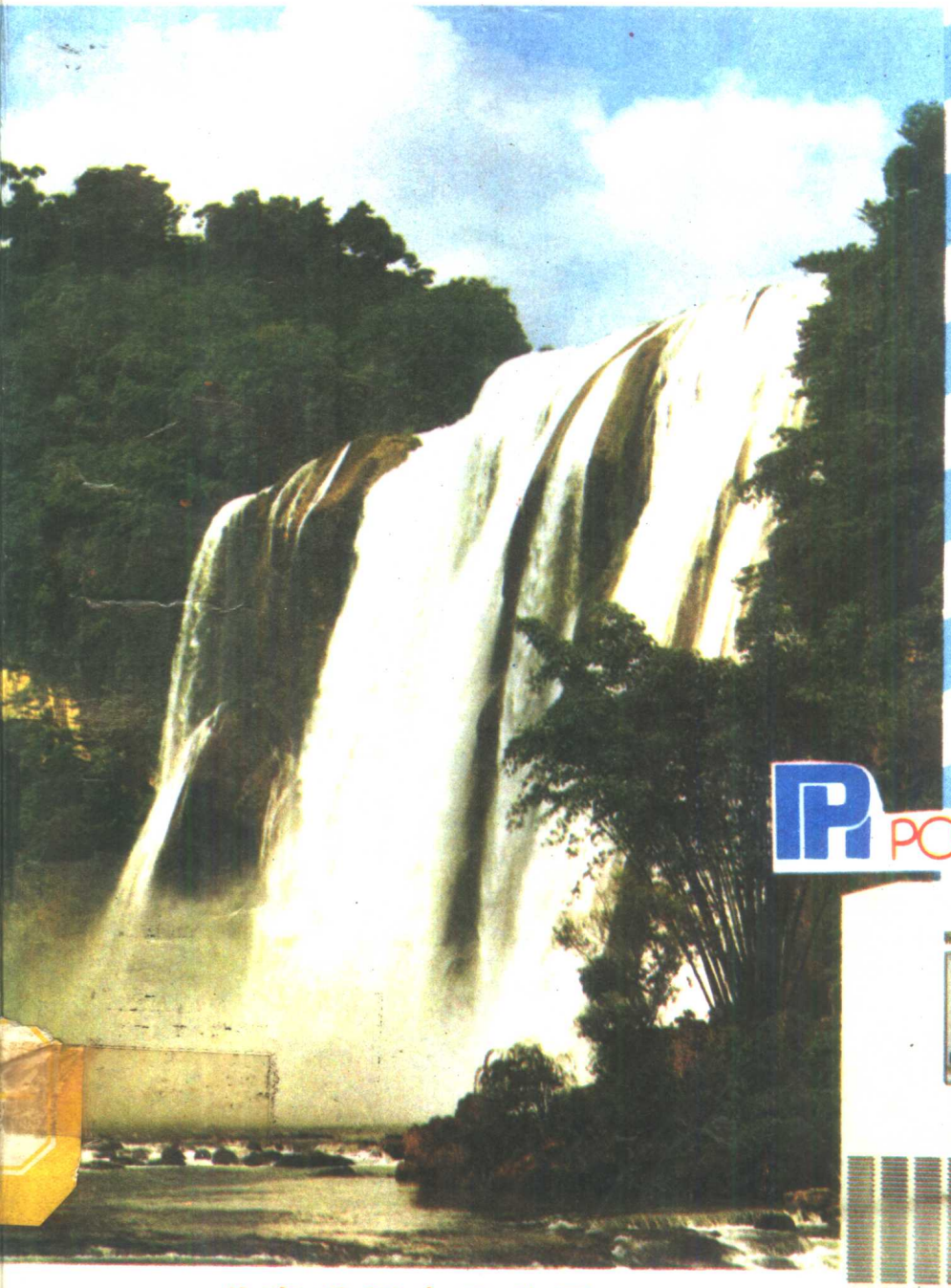
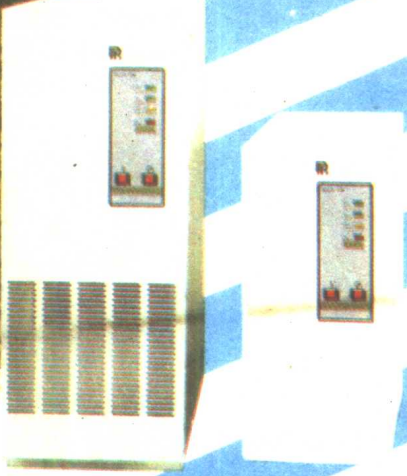
Turbo C++ 图形程序设计

赵唯一　张旗　编著
马建波　审校



POWERHOPE®



北京希望电脑公司

Turbo C++ 图形程序设计

赵唯一　张旗　编著
马　建　波　审校

北京希望电脑公司

内　容　摘　要

本书主要介绍如何用Turbo C++进行图形程序设计。全书共分三个部分。第一部分讨论了Turbo C++对各种视频性能的支持，介绍其新工具的使用方法及如何开发新的功能。第二部分讨论了怎样用第一部分讲述的图形命令来创建图形和实用程序，并演示如何创建各种各样的过程及应用。第三部分讨论了用面向对象的程序设计方法进行图形程序设计，并展示了一些有用的工具程序。

本书要求读者具有一定的C语言基础，适用于需要了解运用计算机图形技术的科技人员，同时也可供高等院校师生使用。

需要本书的用户请与北京8721信箱联系，邮码100080，电话2562329。

Turbo C++ 图形程序设计

赵唯一　张旗　编著

马　建　波　审校

北京希望电脑公司出版（北京市海淀区黄庄）

双青印刷厂印刷

京准印字：3562—91562

内部成本价：18.00元

前　言

目前，计算机人员从事图形程序设计的需求越来越大。在世界各地，大约有八百万台IBM个人计算机和兼容机用于办公和家庭。绝大部分计算机拥有VGA、EGA或CGA的图形功能——越来越多的计算机用户期望程序可以使用图形。这些用户凭经验认识到，图形是帮助理解和掌握手中的软件以及提高编程技术的有效途径。

Turbo C++语言是在Turbo C的基础上发展起来的。它是为适应近代软件工程的发展而产生的。它增加了面向对象的程序设计所需要的抽象数据类型——类，围绕类增加了十种语言机制，并且清除了C语言中宏的不安全性，使程序员编程更方便好用。

本书主要介绍如何用Turbo C++进行图形程序设计。全书共分三个部分。第一部分讨论了Turbo C++对各种视频性能的支持，介绍其新工具的使用方法及如何开发新的功能。第二部分讨论了怎样用第一部分讲述的图形命令来创建图形和实用程序，并演示如何创建各种各样的过程及应用。第三部分讨论了用面向对象的程序设计方法进行图形程序设计，并展示了一些有用的工具程序。

本书是通过编者的精心工作才得以完成的。在编写过程中，我们始终希望将最好的东西以最易懂的方式介绍给广大的读者，使读者能够从中获益。

编者　赵唯一　张　旗

目　录

第一部分　图形程序设计简介

第一批微处理机出现的时候，八位的CPU和总线结构，再带上16K的RAM，就被认为是比较标准的；而且，人们还认为2兆的时钟速度确实够快了，能处理25行80列的视频显示器已非常不错．若是你想进行图形显示，除了熟悉的ASCII字符外，基于ROM的图形字符集就是你唯一的选择．直接对视频存贮器进行存取的手段极少；而且，CPU和系统内存的限制使得可以进行存取的东西难于使用．

这些早期的计算机系统可以分为两类：第一，侧重于高效率地使用内存和极快的处理速度，以便进行科学计算；第二，牺牲了系统的许多性能以支持复杂的单色或彩色图形功能，从而使系统面向所谓的游戏．

在这以后，CPU和操作系统的功能越来越强大，速度越来越快，从而，人们可以更加自由地发挥机器的威力．以Heath/Zenith的Z-100系列（最初的MS-DOS系统）为例，人们可以使用三组视频存贮器，其中每个提供32K或64K的RAM以控制红、蓝、绿颜色枪（总共192K）．如果你的Z-100仅有其中一组（约定是绿色的），则只限于使用单色监视器．倘若你拥有三组视频存贮器和唯一的单色监视器，就可以用灰度的差别模拟彩色．若是有高分辩的彩色监视器，你就可以选择八种彩色（黑、蓝、紫、红、桔、黄、绿和白色）．

更为重要的是，你拥有了可以对象素进行操作的视频存贮器，其显示范围是水平方向640个象素，垂直方向225个象素．对文本而言，显示范围是80个字符列及25行．其实，没有必要对文本方式和图形方式进行区别，二者实质上是一样的．文本字符宽占8个象素，高9个象素（包括了扩展部分，即小写字符，如g，j，p，q扩展到行以下的部分），它们由直接写到视频RAM的ROM字符集创建．

你也可以写（及读）在视频RAM中的个别字符，将它们同标准字符混合起来而不需要改变方式或取消一种显示以支持另外一种显示．

IBM的设计方法是：保持同样的CPU，16位的系统结构及Microsoft DOS，选择最低限度的硬件配置．最早的IBM-PC机配备了相当有限的视频RAM（大约4K），仅适用于非图形方式的显示．若你需要图形能力，彩色图形适配器（CGA）可做为添加物装到机器上，它可提供带有2色（单显）的高分辩率图形（640×200个象素），或者提供带有四个预定义调色板的4种颜色显示的低分辩率图形（320×200）．

改变视频适配器在过去是轻而易举的．一大批一般的开发者看准了RAM片子的价格，兴致极高地从事开发更加高级的视频卡，如Hercules，MCGA，及EGA视频适配器．

视频硬件的增多（现在至少有10种所谓标准一点的视频适配器）给程序员带来了问题．不同的适频硬件设计支持不同的性能，可能使用不同的存贮地址，也许不支持多页图形（也许支持或不支持多页文本），也许支持从320×200到1024×768象素的显示及从单色到256种彩色显示的视频方式．当然，每个图形程序员面临的首要问题就是确定在什么机器上安装了哪种视频适配器．

一种可供选择的办法是询问最终用户，使他们选择将所使用的视频方式．然而，这种方法并不太好，因为就是最熟练的程序员也并不总能确定其所使用的硬件，而且一般的最

终用户也许根本不知道他所用的是图形适配器，更不用说什么硬件或提供的性能了．故而，另一办法就是使软件向硬件进行查询以确定当前的配置．

若是有确认硬件的标准手段存在，则这种方法就很简单．然而，对程序员来讲，虽然适配器硬件迅速增多，却没有正规的确认手段．解决这一问题的办法如设计一些测试，以确定CGA/EGA视频适配器有无．

令人欣慰的是，Turbo C++（及 Turbo Pascal5.5 ）大大消除了在确认和支持及使用当前视频适配器卡方面的不确定性．而且，从Borland软件过去的成绩看， 可以相当有把握地讲将来新的视频适配器也会受到类似的支持．

然而，对各种视频性能的这种支持也容易使人在下面几个方面搞不清楚：如何使用这些新工具，哪些是可能的，如何开发那些新的功能．

消除这种混乱正是本书要研究的内容．

第 1 章　图形视频适配器

Turbo C++和Turbo Pascal都为当前使用的主要视频卡类型提供了全面的支持。本章将仅仅讨论和举例以说明Turbo C++。如果你使用Turbo Pascal，相应的函数也有，只要使用同样的函数和过程名且以同样的方式进行操作就行。

当前的视频卡，其种类很广，包括最早只适用文本的视频系统及超高分辨率的 IBM-8514（在排版的CAD软件中比较流行）。显然，高的分辨率视频卡必须同有相应象素分辨率的监视器匹配。然而，这是个硬件问题，你只需假设你的硬件对应于 Turbo C++ 中 detectgraph函数返回的类型。视频卡和监视器之间的不一致是最终用户的责任，程序员不必关心。

1.1　视频方式

每一台PC，XT，AT或PS/2都配备了某种视频适配器卡。从基本的卡开始，你可能有一个单色显示适配器（MDA），它只支持文本显示。若是这样，不将系统提高档次你就不能进行图形程序设计。

第一个档次就是流行的彩色图形适配器卡（CGA）。Hercules单色图形适配器，多彩色图形矩阵（MCGA），增强型图形适配器（EGA）提供了更高的分辨率和更宽的选择。对更高级的图形性能，如排版和CAD中的应用，AT&T（400行图形适配器），可变的或视频的图形阵列（VGA），PC-3270及IBM-8514视频适配器都提供了更高的象素分辨率及颜色选择。

在Turbo C++中，对上述所有适配器的支持形式是：6种图形接口（.BGI）单元（ATT CGA，EGAVGA，HERC，IBM-8514和PC-3270）和4种图形字体（GOTH.CHR，LITT.CHR，SANS.CHR及TRIP.CHR）。随着新的视频图形卡的出现，新的.BGI单元有可能包括进来用于支持。然而，新的.CHR字体可能由用户自己创造。在标准的存贮模式中，没有图形支持单元，其目的是在图形不需要时提高编译速度。要使用图形模块，有二个方法。

第一，Borland提供了TLIB，它是Turbo C++中的Turbo库管理程序。使用TLIB，在一个或多个存贮模式库中就可以包括进图型库（GRAPHICS.LIB）。关于TLIB的使用在第九章中有详细的解释。

第二，可以为每个程序创建一个工程文件（.PRJ），使GRAPHICS.LIB可被连接和使你的程序在需要时能从磁盘上装入合适的.BGI文件。

如果你要进行大量的图形程序设计，建议你将GRAPHICS.LIB 包括在你的标准库中。BGIOBJ实用程序也可用来把.BGI图形驱动程序和.CHR字体文件转换为目标文件，使你能将它们直接连接到你的程序里（因此将它们直接装进你的.EXE程序）。

然而，就测试和开发而言，GRAPHICS.LIB常被称为辅助库，而且，.BGI和.CHR文件可从外部获得。

1.2　连进图形库

如果你在使用Turbo C++行命令编译器（TCC.EXE）去编译称之为YOURPROG.C的一个源程序，TCC命令是：

```
tcc yourprog graphics.lib
```

这么做已经假定YOURPROG.C和GRAPHICS.LIB同TCC在相同目录里，因而不需要路径说明.

在使用Turbo C++集成编译器（TC.EXE）时，你需要创建和选择一个工程文件以指示连接程序去取得外部的库（GRAPHICS.LIB）及标准库．这个工程文件很简单，就本例而言，只需一行：

```
yourprog graphics.lib
```

这里，如果GRAPHICS.LIB装在一个不同的子目录中，你或者可以在YOURPROG.PRJ中说明路径，或者在编译器环境配置中包括这条路径（使用Alt+O，然后选择环境）．对这两种情况的任何一个而言，在选择RUN可选项之前或者在将源程序编译到磁盘上之前，你都必须选键入工程文件名YOURPROG.PRJ（PRJ扩展名是可选的，若不指明，就认为是它）．

注意，缺少库函数并不会在编译期间生成一个错误信息——只是在第二遍连接时才出现．只要编译程序发现图形文件在GRAPHICS.H头文件中被正确地定义（在源代码中一定要使用#include <graphics.h>），在连接程序发现其自身不能获取库之前，不会产生错误信息.

为了说明一个PRJ文件，输入Alt+P，弹出Project菜单．选择第一个菜单项（Open project...），或输入工程文件名，或用鼠标或箭头键选取正确的.PRJ文件.

记住，如果你已经将TC设置成自动保存，则选定的工程文件在你下次使用C++时可自动被选定．关闭工程（Close project）选择项可被用来取消这一选择，使你可以编译和运行另一个程序．另外，在Turbo C++的查找表中可以保存几个不同的工程文件.

缺点

一旦一个程序被编译过（从一个工程文件或使用命令行的说明），.EXE文件连同所必需的外部的.BGI和.CHR文件就可以被使用．这些文件需要大约60K磁盘空间，从存贮角度看这不算过分的要求．但是，执行时依赖外部文件会导致一些问题.

首先，对initgraph的调用必须包括关于.BGI（及.CHR）模块在何处的驱动器/路径的说明．若不指定路径，就假定为当前目录．这一常用的信息通常由程序员提供．而且，若所需文件没有找到则程序必定终止.

第二，如果约定的（即当前的）路径被指定且所有外部文件都在，但程序是从另一个目录或驱动器调用的，也将会出现程序终止.

第三，别指望最终用户会意识到外部文件的重要性．他们可能非常喜爱你的程序并认真地保护它，但当空间不够时，有可能在不知道的情况下删掉必要的外部文件.

1.3 最初的工作

图形程序的第一步是将相应的图形驱动程序初始化。表1-1列出了Turbo C++支持的一些图形视频卡、驱动程序和图形方式。

Graphics[1] Driver Constant	Number Value	Graphics Mode(s)	Key[2] Value	Column x Row	Palette[3] or Colors	Video Pages
DETECT	0	requests initgraph to execute autodetection				
CGA	1	CGAC0	0	320 x 200	C0	1
		CGAC1	1	320 x 200	C1	1
		CGAC2	2	320 x 200	C2	1
		CGAC3	3	320 x 200	C3	1
		CGAHI	4	640 x 200	2 colors	1
MCGA	2	MCGAC0	0	320 x 200	C0	1
		MCGAC1	1	320 x 200	C1	1
		MCGAC2	2	320 x 200	C2	1
		MCGAC3	3	320 x 200	C3	1
		MCGAMED	4	640 x 200	2 colors	1
		MCGAHI	5	640 x 480	2 colors	1
EGA	3	EGALO	0	640 x 200	16 colors	4
		EGAHI	1	640 x 350	16 colors	2
EGA64	4	EGA64LO	0	640 x 200	16 colors	1
		EGA64HI	1	640 x 350	4 colors	1
EGAMONO	5	EGAMONOHI	3	640 x 350	2 colors	1-2[4]
IBM8514[5]	6	IBM8514LO	0	640 x 480	256 colors	1
		IBM8514HI	1	1024 x 768	256 colors	1
HERC	7	HERCMONOHI	0	720 x 348	2 colors	4
ATT400	8	ATT400C0	0	320 x 200	C0	1
		ATT400C1	1	320 x 200	C1	1
		ATT400C2	2	320 x 200	C2	1
		ATT400C3	3	320 x 200	C3	1
		ATT400MED	4	640 x 200	2 colors	1
		ATT400HI	5	640 x 400	2 colors	1
VGA	9	VGALO	0	640 x 200	16 colors	4
		VGAMED	1	640 x 350	16 colors	2
		VGAHI	2	640 x 480	16 colors	1
PC3270	10	PC3270HI	0	720 x 350	2 colors	1

表1-1：视频方式

注1. graphic_drivers和graphic_modes的名字是定义在GRAPHICS.H中的常数， 相应的数

字值和方式值也如此（见2）。
2. 方式设置由initgraph，detectgraph，或getgraphmode返回。
3. CGAC3指预定义的4种调色板。参见setpalette。
4. 对有64K的已EGAMONO卡，仅支持一个视频页；对256K而言，支持2个视频页。
5. Autodetection不会正确地辨认出IBM-8514图形卡。相反，initgraph或detectgraph将把IBM-8514卡看成一个VGA图形卡，而对这个卡IBM-8514将会正确地模仿（IBM-8514LO 等价于VGAHI）。采用高分辨方式（IBM-8514HI，1024×768个象素），在调用initgraph前，置值IBM8514（数字值6，在GRAPHICS.H中定义）给变量graphdriver。不要把detectgraph或DETECT同initgraph一起使用。参见关于IBM-8514的注释及setrbgpalette。

detectgraph

通常，detectgraph函数受initgraph调用，但它也可独立地被使用。例：

```
#include <graphics.h>
int graphdriver = DETECT, graphmade;
main
{
        detectgraph( &graphdriver, &graphmode );
        .......
}
```

若出现问题，*graphdriver返回一个错误代码；否则，*graphdrioer确定出正确的驱动程序类型，*graphmode返回这个驱动程序最高的且有效的视频方式。

对detectgraph，不要求驱动程序路径（参考initgraph）。

detectgraph函数并不初始化任何图形设置。直接调用 detectgraph ， 然后或使用initgraph调用一个特定的驱动程序，或选择initgraph不会用约定方式调用的一种图形方式。在用setgraphmode函数将图形初始化之后，可以调用不同的方式。

返回值：*graphdriver返回驱动程序类型或错误代码；*graphmode返回最高且有效的视频方式。

可移植性：IBM-PC机及兼容机。在Turbo Pascal中也有相应的函数。

initgraph

initgraph函数用来设置初始的图形参数值，装入合适的图形驱动程序， 将工程设置成所希望的图形方式。例：

```
#include <graphics.h>
int graphdriver = DETECT, graphmode;
char driverpath = "";
main
```

```
{
        initgraph( &graphdriver, &graphmode,
                   driverpath );
        ......
}
```

将graphdriver置为0就是指示initgraph调用detectgraph (graphdriver, graphmode) 去确定所安装的图形适配器的类型（和设置）。如果出现错误，*graphdriver返回表示错误类型的错误代码，如表1-2所示。

Error Code	Meaning
-2	Cannot detect graphics card
-3	Cannot locate graphics driver file(s)
-4	Invalid driver (or not recognized)
-5	Insufficient memory to load graphics driver

表1-2：初始化图形错误代码

detectgraph和graphresult函数返回同样的错误代码。若没有出现错误，则内部的错误代码被置为0。initgraph为正确的图形驱动程序分配内存，从磁盘装入必需的.BGI文件，设置约认的图形参数值。进一步，*graphdriver返回驱动程序类型，而*graphmode返回方式设置。

另外，*graphdriver和*graphmode可以用正确的数字常数或在<GRAPHICS.H>中定义的驱动程序和方式名来说明。对任何一种情况而言，driverpath表明.BGI图形驱动程序所处位置的驱动器和路径。如果driverpath为空值（如上例中所示），则这些文件必须放在约定的目录中。如果它们被放在另外的目录里，则整个路径说明必须象如下一样：

```
char driverpath = "\\TURBOC\\DRIVERS";
```

注意两个反斜线的使用（\\）。反斜线符被用来设置脱字序列（如在表示换行的\n和表示响铃的\a中那样）。要在字符串中写进一个反斜线，需要写成\\。

由initgraph设置的driverpath也被settextstyle用来搜索字符字体文件（.CHR）。.BGI和.CHR都必须在同一个目录中。

返回值：*graphdriver返回驱动程序类型或错误代码；*graphmode返回最高且有效的视频方式。

可移植性：IBM-PC机及兼容机。在Turbo pascal中也有相应的函数。

1.4 有关图形出错的函数

如果图形视频卡没有，则图形驱动程序就不会找到。如果在初始化检测中出现某一其它错误，detectgraph或initgraph就会返回错误代码。但是，也有这样的情况：图形错误

发生了，可用graphresult和grapherrormsg来检测和显示相应的错误结果和信息。

graphresult

graphresult函数返回导致错误的最后的图形操作设置的数字错误代码。它将是个在范围-18至0之间的整数值。由于调用gtaphresult时，错误情况被复置为0，返回值应被存在一个局部变量中，然后做为下步工作的测试之用。例：

```
#include <graphics.h>
int errornumber;
errornumber = graphresult();
```

返回值：错误代码（-15，0），见表1-3中对图形错误信息的解释。
可移植性：IBM-PC机及兼容机。在Turbo Pascal中也有相应的函数。

Error Code	Graphics Error Constant[1]	Corresponding Error Message String
0	grOk	No error
-1	grNoInitGraph	(BGI) graphics not installed (use initgraph)
-2	grNotDetected	Graphics hardware not detected
-3	grFileNotFound	Device driver file not found (.BGI file)
-4	grInvalidDriver	Invalid device driver file
-5	grNoLoadMem	Not enough memory to load driver
-6	grNoScanMem	Out of memory in scan fill
-7	grNoFloodMem	Out of memory in flood fill
-8	grFontNotFound	Font file not found (.CHR file)
-9	grNoFontMem	Not enough memory to load font
-10	grInvalidMode	Invalid graphics mode for selected driver
-11	grError	Graphics error (generic error)
-12	grIOerror	Graphics I/O error
-13	grInvalidFont	Invalid font file
-14	grInvalidFontNum	Invalid font number
-15	grInvalidDeviceNum	Invalid device number
-18	grInvalidVersion	Invalid version number

表1-3：图形错误信息

注1. graphics_error常数和错误信息定义在GRAPHICS.H中，错误值-16..-17没有采用。

grapherrormsg

grapherrormsg函数返回一个指向相应错误信息字符串的指针。这些串定义在图形库（GRAPHICS.LIB）中，但可以创建一个单独的错误信息例程以显示一条可使人获得更多情况的错误信息。例：

```
#include <graphics.h>
int errornumber;
errornumber = graphresult();
printf(" %s ", grapherrormsg( errornumber ) );
```

返回值：无
可移植性：IBM-PC机及兼容机。在Turbo Pascal中也有相应的函数。

1.5 其它图形方式函数

除了EGAMONO，HERC和PC-3270视频驱动程序属于例外以外，每一个视频驱动程序支持两种或更多的视频方式，这些方式提供了各不相同的象素分辨率或不同的颜色调色板。为了处理方式查询和改变操作方式，Turbo C++提供了好几个函数。

getgraphmode

getgraphmode 函数返回表明当前图形方式的一个整数值，这个值由 initgraph 或 setgraphmode设置。例：

```
#include <graphics.h>
int currentmode;
currentmode = getgraphmode();
```

返回值：当前图形方式。
可移值性：IBM-PC机及兼容机。在Turbo Pascal中也有相应的函数。

getmoderange

getmoderange函数被调用时带有一个指明图形驱动程序的整数值。它可以是整型变量或在GRAPHICS.H中定义的一个常数。这个函数返回两个值，分别定义了所指驱动程序的最小和最大有效方式。例：

```
#include <graphics.h>
int lomode,himode;
getmoderange( graphdriver, &lomode, &himode );
```

如果做为graphdriver传送的值不合法，则lomode和himode都返回-1。
返回值：最大或最小合法方式或错误代码-1
可移植性：IBM-PC机及兼容机。在Turbo Pascal中也有相应的函数。