C语言图形函数介绍

Turbo C提供了非常丰富的图形函数, 所有图形函数的原型均在graphics. h中, 这里主要介绍图形模式的初始化、独立图形程序的建立、基本图形功能、图形窗口以及图形模式下的文本输出等函数。另外, 使用图形函数时要确保有显示器图形驱动程序\*BGI, 同时将集成开发环境Options/Linker中的Graphics lib选为on, 只有这样才能保证正确使用图形函数。

1. 图形模式的初始化

不同的显示器适配器有不同的图形分辨率。即是同一显示器适配器, 在不同模式下也有不同分辨率。因此, 在屏幕作图之前, 必须根据显示器适配器种类将显示器设置成为某种图形模式, 在未设置图形模式之前, 微机系统默认屏幕为文本模式(80列, 25行字符模式), 此时所有图形函数均不能工作。设置屏幕为图形模式, 可用下列图形初始化函数:

void far initgraph(int far \*gdriver, int far \*gmode, char \*path);

其中gdriver和gmode分别表示图形驱动器和模式, path是指图形驱动程序所在的目录路径。有关图形驱动器、图形模式的符号常数及对应的分辨率见表2。

图形驱动程序由Turbo C出版商提供, 文件扩展名为.BGI。根据不同的图形适配器有不同的图形驱动程序。例如对于EGA、 VGA 图形适配器就调用驱动程序 EGAVGA.BGI。

表2. 图形驱动器、模式的符号常数及数值

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

图形驱动器(gdriver) 图形模式(gmode)

─────────── ─────────── 色调 分辨率

符号常数 数值 符号常数 数值

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

VGA 9 VGALO 0 16色 640\*200

VGAMED 1 16色 640\*350

VGAHI 2 16色 640\*480

───────────────────────────────────

PC3270 10 PC3270HI 0 2色 720\*350

───────────────────────────────────

DETECT 0 用于硬件测试

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

表2. 图形驱动器、模式的符号常数及数值  
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  
    图形驱动器(gdriver)    图形模式(gmode)  
   ───────────     ───────────    色调  分辨率  
   符号常数    数值       符号常数     数值  
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  
    CGA          1            CGAC0           0          C0   320\*200  
                              CGAC1           1          C1   320\*200  
                              CGAC2           2          C2   320\*200  
                              CGAC3           3          C3   320\*200  
                              CGAHI           4          2色  640\*200  
───────────────────────────────────  
    MCGA         2           MCGAC0         0          C0   320\*200  
                              MCGAC1          1          C1   320\*200  
                              MCGAC2          2          C2   320\*200  
                              MCGAC3          3          C3   320\*200  
                              MCGAMED       4          2色  640\*200  
                              MCGAHI          5          2色  640\*480  
───────────────────────────────────  
    EGA          3            EGALO           0         16色  640\*200  
                              EGAHI           1         16色  640\*350  
───────────────────────────────────  
    EGA64        4            EGA64LO         0         16色  640\*200  
                              EGA64HI         1          4色  640\*350  
───────────────────────────────────  
    EGAMON       5         EGAMONHI       0          2色  640\*350  
───────────────────────────────────  
    IBM8514      6            IBM8514LO       0        256色  640\*480  
                              IBM8514HI       1        256色 1024\*768  
───────────────────────────────────  
    HERC         7            HERCMONOHI    0          2色  720\*348  
───────────────────────────────────  
    ATT400       8            ATT400C0        0          C0   320\*200  
                              ATT400C1        1          C1   320\*200  
                              ATT400C2        2          C2   320\*200  
                              ATT400C3        3          C3   320\*200  
                              ATT400MED     4          2色  320\*200  
                              ATT400HI        5          2色  320\*200  
───────────────────────────────────  
    VGA          9            VGALO           0         16色  640\*200  
                              VGAMED          1         16色  640\*350  
                              VGAHI           2         16色  640\*480  
───────────────────────────────────  
    PC3270       10           PC3270HI        0          2色  720\*350  
───────────────────────────────────  
    DETECT       0            用于硬件测试

例4. 使用图形初始化函数设置VGA高分辨率图形模式

#include <graphics.h>

int main()

{

int gdriver, gmode;

gdriver=VGA;

gmode=VGAHI;

initgraph(&gdriver, &gmode, "c:\\tc");

bar3d(100, 100, 300, 250, 50, 1); /\*画一长方体\*/

getch();

closegraph();

return 0;

}

有时编程者并不知道所用的图形显示器适配器种类, 或者需要将编写的程序 用于不同图形驱动器, Turbo C提供了一个自动检测显示器硬件的函数, 其调用

格式为:

void far detectgraph(int \*gdriver, \*gmode);

其中gdriver和gmode的意义与上面相同。

例5. 自动进行硬件测试后进行图形初始化

#include <graphics.h>

int main()

{

int gdriver, gmode;

detectgraph(&gdriver, &gmode); /\*自动测试硬件\*/

printf("the graphics driver is %d, mode is %d\n", gdriver, gmode); /\*输出测试结果\*/

getch();

initgraph(&gdriver, &gmode, "c:\\tc");

/\* 根据测试结果初始化图形\*/

bar3d(10, 10, 130, 250, 20, 1);

getch();

closegraph();

return 0;

}

上例程序中先对图形显示器自动检测, 然后再用图形初始化函数进行初始化设置, 但Turbo C提供了一种更简单的方法, 即用gdriver= DETECT 语句后再跟 initgraph()函数就行了。采用这种方法后, 上例可改为:

例6.

#include <graphics.h>

int main()

{

int gdriver=DETECT, gmode;

initgraph(&gdriver, &gmode, "c:\\tc");

bar3d(50, 50, 150, 30, 1);

getch();

closegraph();

return 0;

}

另外, Turbo C提供了退出图形状态的函数closegraph(), 其调用格式为:void far closegraph(void);调用该函数后可退出图形状态而进入文本方式(Turbo C 默认方式), 并释放用于保存图形驱动程序和字体的系统内存。

2. 独立图形运行程序的建立

Turbo C对于用initgraph()函数直接进行的图形初始化程序, 在编译和链接时并没有将相应的驱动程序(\*.BGI)装入到执行程序, 当程序进行到intitgraph()语句时, 再从该函数中第三个形式参数char \*path中所规定的路径中去找相应的驱动程序。若没有驱动程序, 则在C:\TC中去找, 如C:\TC中仍没有或TC不存在,将会出现错误:

BGI Error: Graphics not initialized (use 'initgraph')

因此,为了使用方便,应该建立一个不需要驱动程序就能独立运行的可执行图形程序,Turbo C中规定用下述步骤(这里以EGA、VGA显示器为例):

1. 在C:\TC子目录下输入命令:BGIOBJ EGAVGA

此命令将驱动程序EGAVGA.BGI转换成EGAVGA.OBJ的目标文件。

2. 在C:\TC子目录下输入命令:TLIB LIB\GRAPHICS.LIB+EGAVGA

此命令的意思是将EGAVGA.OBJ的目标模块装到GRAPHICS.LIB库文件中。

3. 在程序中initgraph()函数调用之前加上一句:

registerbgidriver(EGAVGA\_driver):

该函数告诉连接程序在连接时把EGAVGA的驱动程序装入到用户的执行程序中。

经过上面处理,编译链接后的执行程序可在任何目录或其它兼容机上运行。假设已作了前两个步骤,若再向例6中加 registerbgidriver()函数则变成:

例7:

#include<stdio.h>

#include<graphics.h>

int main()

{

int gdriver=DETECT,gmode;

registerbgidriver(EGAVGA\_driver): / \*建立独立图形运行程序 \*/

initgraph( gdriver, gmode,"c:\\tc");

bar3d(50,50,250,150,20,1);

getch();

closegraph();

return 0;

}

上例编译链接后产生的执行程序可独立运行。

如不初始化成EGA或CGA分辨率, 而想初始化为CGA分辨率, 则只需要将上述步骤中有EGAVGA的地方用CGA代替即可。

3.屏幕颜色的设置和清屏函数

对于图形模式的屏幕颜色设置, 同样分为背景色的设置和前景色的设置。在Turbo C中分别用下面两个函数。

设置背景色: void far setbkcolor( int color);

设置作图色: void far setcolor(int color);

其中color 为图形方式下颜色的规定数值, 对EGA, VGA显示器适配器, 有关颜色的符号常数及数值见下表所示。

表3 有关屏幕颜色的符号常数表

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

符号常数 数值 含义 符号常数 数值 含义

───────────────────────────────────

BLACK 0 黑色 DARKGRAY 8 深灰

BLUE 1 兰色 LIGHTBLUE 9 深兰

GREEN 2 绿色 LIGHTGREEN 10 淡绿

CYAN 3 青色 LIGHTCYAN 11 淡青

RED 4 红色 LIGHTRED 12 淡红

MAGENTA 5 洋红 LIGHTMAGENTA 13 淡洋红

BROWN 6 棕色 YELLOW 14 黄色

LIGHTGRAY 7 淡灰 WHITE 15 白色

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

清除图形屏幕内容使用清屏函数, 其调用格式如下:

voide far cleardevice(void);

有关颜色设置、清屏函数的使用请看例8。

例8:

#include<stdio.h>

#include<graphics.h>

int main()

{

int gdriver, gmode, i;

gdriver=DETECT;

registerbgidriver(EGAVGA\_DRIVER);/\*建立独立图形运行程序\*/

initgraph(&gdriver, &gmode", "");/\*图形初始化\*/

setbkcolor(0); /\*设置图形背景\*/

cleardevice();

for(i=0; i<=15; i++)

{

setcolor(i); /\*设置不同作图色\*/

circle(320, 240, 20+i\*10); /\*画半径不同的圆\*/

delay(100); /\*延迟100毫秒\*/

}

for(i=0; i<=15; i++)

{

setbkcolor(i); /\*设置不同背景色\*/

cleardevice();

circle(320, 240, 20+i\*10);

delay(100);

}

closegraph();

return 0;

}

另外, TURBO C也提供了几个获得现行颜色设置情况的函数。

int far getbkcolor(void); 返回现行背景颜色值。

int far getcolor(void); 返回现行作图颜色值。

int far getmaxcolor(void); 返回最高可用的颜色值。

4. 基本图形函数

基本图形函数包括画点, 线以及其它一些基本图形的函数。本节对这些函数作一全面的介绍。

一、画点

1. 画点函数

void far putpixel(int x, int y, int color);

该函数表示有指定的象元画一个按color所确定颜色的点。对于颜色color的值可从表3中获得而对x, y是指图形象元的坐标。

在图形模式下, 是按象元来定义坐标的。对VGA适配器, 它的最高分辨率为640x480, 其中640为整个屏幕从左到右所有象元的个数, 480 为整个屏幕从上到下所有象元的个数。屏幕的左上角坐标为(0, 0), 右下角坐标为(639, 479), 水平方向从左到右为x轴正向, 垂直方向从上到下为y轴正向。TURBO C 的图形函数 都是相对于图形屏幕坐标, 即象元来说的。

关于点的另外一个函数是:

int far getpixel(int x, int y);

它获得当前点(x, y)的颜色值。

2. 有关坐标位置的函数

int far getmaxx(void);

返回x轴的最大值。

int far getmaxy(void);

返回y轴的最大值。

int far getx(void);

返回游标在x轴的位置。

void far gety(void);

返回游标有y轴的位置。

void far moveto(int x, int y);

移动游标到(x, y)点, 不是画点, 在移动过程中亦画点。

void far moverel(int dx, int dy);

移动游标从现行位置(x, y)移动到(x+dx, y+dy)的位置, 移动过程中不画点。

二、画线

1. 画线函数

TURBO C提供了一系列画线函数, 下面分别叙述:

void far line(int x0, int y0, int x1, int y1);

画一条从点(x0, y0)到(x1, y1)的直线。

void far lineto(int x, int y);

画一作从现行游标到点(x, y)的直线。

void far linerel(int dx, int dy);

画一条从现行游标(x, y)到按相对增量确定的点(x+dx, y+dy)的直线。

void far circle(int x, int y, int radius);

以(x, y)为圆心, radius为半径, 画一个圆。

void far arc(int x, int y, int stangle, int endangle, int radius);

以(x, y)为圆心, radius为半径, 从stangle开始到endangle结束(用度表示)画一段圆弧线。在TURBO C中规定x轴正向为0度, 逆时针方向旋转一周, 依次为90, 180, 270和360度(其它有关函数也按此规定, 不再重述)。

void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

以(x, y)为中心, xradius, yradius为x轴和y轴半径, 从角stangle 开始到endangle结束画一段椭圆线, 当stangle=0, endangle=360时, 画出一个完整的椭圆。

void far rectangle(int x1, int y1, int x2, inty2);

以(x1, y1)为左上角, (x2, y2)为右下角画一个矩形框。

void far drawpoly(int numpoints, int far \*polypoints);

画一个顶点数为numpoints, 各顶点坐标由polypoints 给出的多边形。polypoints整型数组必须至少有2倍顶点数个无素。每一个顶点的坐标都定义为x, y, 并且x在前。值得注意的是当画一个封闭的多边形时, numpoints 的值取实际多边形的顶点数加一, 并且数组polypoints中第一个和最后一个点的坐标相同。

下面举一个用drawpoly()函数画箭头的例子。

例9:

#include<stdlib.h>

#include<graphics.h>

int main()

{

int gdriver, gmode, i;

int arw[16]={200, 102, 300, 102, 300, 107, 330,

100, 300, 93, 300, 98, 200, 98, 200, 102};

gdriver=DETECT;

registerbgidriver(EGAVGA\_driver);

initgraph(&gdriver, &gmode, "");

setbkcolor(BLUE);

cleardevice();

setcolor(12); /\*设置作图颜色\*/

drawpoly(8, arw); /\*画一箭头\*/

getch();

closegraph();

return 0;

}

2. 设定线型函数

在没有对线的特性进行设定之前, TURBO C用其默认值, 即一点宽的实线, 但TURBO C也提供了可以改变线型的函数。线型包括:宽度和形状。其中宽度只有两种选择: 一点宽和三点宽。而线的形状则有五种。下面介绍有关线型的设置函数。

void far setlinestyle(int linestyle, unsigned upattern, int thickness);

该函数用来设置线的有关信息, 其中linestyle是线形状的规定, 见表5。

表5. 有关线的形状(linestyle)

━━━━━━━━━━━━━━━━━━━━━━━━━

符号常数 数值 含义

─────────────────────────

SOLID\_LINE 0 实线

DOTTED\_LINE 1 点线

CENTER\_LINE 2 中心线

DASHED\_LINE 3 点画线

USERBIT\_LINE 4 用户定义线

━━━━━━━━━━━━━━━━━━━━━━━━━

thickness是线的宽度, 见表6。

表6. 有关线宽(thickness)

━━━━━━━━━━━━━━━━━━━━━━━━━

符号常数 数值 含义

─────────────────────────

NORM\_WIDTH 1 一点宽

THIC\_WIDTH 3 三点宽

━━━━━━━━━━━━━━━━━━━━━━━━━

对于upattern, 只有linestyle选USERBIT\_LINE 时才有意义( 选其它线型, uppattern取0即可)。此进uppattern的16位二进制数的每一位代表一个象元, 如果那位为1, 则该象元打开, 否则该象元关闭。

void far getlinesettings(struct linesettingstype far \*lineinfo);

该函数将有关线的信息存放到由lineinfo 指向的结构中, 表中linesettingstype的结构如下:

struct linesettingstype{

int linestyle;

unsigned upattern;

int thickness;

}

例如下面两句程序可以读出当前线的特性

struct linesettingstype \*info;

getlinesettings(info);

void far setwritemode(int mode);

该函数规定画线的方式。如果mode=0, 则表示画线时将所画位置的原来信息覆盖了(这是TURBO C的默认方式)。如果mode=1, 则表示画线时用现在特性的线与所画之处原有的线进行异或(XOR)操作, 实际上画出的线是原有线与现在规定的线进行异或后的结果。因此, 当线的特性不变, 进行两次画线操作相当于没有画线。

有关线型设定和画线函数的例子如下所示。

例10.

#include<stdlib.h>

#include<graphics.h>

int main()

{

int gdriver, gmode, i;

gdriver=DETECT;

registerbgidriver(EGAVGA\_driver);

initgraph(&gdriver, &gmode, "");

setbkcolor(BLUE);

cleardevice();

setcolor(GREEN);

circle(320, 240, 98);

setlinestyle(0, 0, 3); /\*设置三点宽实线\*/

setcolor(2);

rectangle(220, 140, 420, 340);

setcolor(WHITE);

setlinestyle(4, 0xaaaa, 1); /\*设置一点宽用户定义线\*/

line(220, 240, 420, 240);

line(320, 140, 320, 340);

getch();

closegraph();

return 0;

}

5. 封闭图形的填充

填充就是用规定的颜色和图模填满一个封闭图形。

一、先画轮廓再填充

TURBO C提供了一些先画出基本图形轮廓, 再按规定图模和颜色填充整个封闭图形的函数。在没有改变填充方式时, TURBO C以默认方式填充。 下面介绍这些函数。

void far bar(int x1, int y1, int x2, int y2);

确定一个以(x1, y1)为左上角, (x2, y2)为右下角的矩形窗口, 再按规定图模和颜色填充。

说明: 此函数不画出边框, 所以填充色为边框。

void far bar3d(int x1, int y1, int x2, int y2, int depth, int topflag);

当topflag为非0时, 画出一个三维的长方体。当topflag为0时, 三维图形不封顶, 实际上很少这样使用。

说明: bar3d()函数中, 长方体第三维的方向不随任何参数而变, 即始终为45度的方向。

void far pieslice(int x, int y, int stangle, int endangle, int radius);

画一个以(x, y)为圆心, radius为半径, stangle为起始角度, endangle 为终止角度的扇形, 再按规定方式填充。当stangle=0, endangle=360 时变成一个实心圆, 并在圆内从圆点沿X轴正向画一条半径。

void far sector(int x, int y, int stanle, intendangle, int xradius, int yradius);

画一个以(x, y)为圆心分别以xradius, yradius为x轴和y轴半径, stangle 为起始角, endangle为终止角的椭圆扇形, 再按规定方式填充。

二、设定填充方式

TURBO C有四个与填充方式有关的函数。下面分别介绍:

void far setfillstyle(int pattern, int color);

color的值是当前屏幕图形模式时颜色的有效值。pattern的值及与其等价的符号常数如表7所示。

表7. 关于填充式样pattern的规定

━━━━━━━━━━━━━━━━━━━━━━━━━━━

符号常数 数值 含义

───────────────────────────

EMPTY\_FILL 0 以背景颜色填充

SOLID\_FILL 1 以实填充

LINE\_FILL 2 以直线填充

LTSLASH\_FILL 3 以斜线填充(阴影线)

SLASH\_FILL 4 以粗斜线填充(粗阴影线)

BKSLASH\_FILL 5 以粗反斜线填充(粗阴影线)

LTBKSLASH\_FILL 6 以反斜线填充(阴影线)

HATCH\_FILL 7 以直方网格填充

XHATCH\_FILL 8 以斜网格填充

INTTERLEAVE\_FILL 9 以间隔点填充

WIDE\_DOT\_FILL 10 以稀疏点填充

CLOSE\_DOS\_FILL 11 以密集点填充

USER\_FILL 12 以用户定义式样填充

━━━━━━━━━━━━━━━━━━━━━━━━━━━━

除USER\_FILL(用户定义填充式样)以外, 其它填充式样均可由setfillstyle() 函数设置。当选用USER\_FILL时, 该函数对填充图模和颜色不作任何改变。 之所以定义USER\_FILL主要因为在获得有关填充信息时用到此项。

void far setfillpattern(char \* upattern,int color);

设置用户定义的填充图模的颜色以供对封闭图形填充。

其中upattern是一个指向8个字节的指针。这8个字节定义了8x8点阵的图形。每个字节的8位二进制数表示水平8点, 8个字节表示8行, 然后以此为模型向个封闭区域填充。

void far getfillpattern(char \* upattern);

该函数将用户定义的填充图模存入upattern指针指向的内存区域。

void far getfillsetings(struct fillsettingstype far \* fillinfo);

获得现行图模的颜色并将存入结构指针变量fillinfo中。其中fillsettingstype结构定义如下:

struct fillsettingstype{

int pattern; /\* 现行填充模式 \* /

int color; /\* 现行填充模式 \* /

};

有关图形填充图模的颜色的选择, 请看下面例程。

例11:

#include<graphics.h>

main(){

char str[8]={10,20,30,40,50,60,70,80}; /\*用户定义图模\*/

int gdriver,gmode,i;

struct fillsettingstype save; /\*定义一个用来存储填充信息的结构变量\*/

gdriver=DETECT;

initgraph(&gdriver,&gmode,"c:\\tc");

setbkcolor(BLUE);

cleardevice();

for(i=0;i<13;i++)

{

setcolor(i+3);

setfillstyle(i,2+i); /\* 设置填充类型 \*

bar(100,150,200,50); /\*画矩形并填充\*/

bar3d(300,100,500,200,70,1); /\* 画长方体并填充\*/

pieslice(200, 300, 90, 180, 90);/\*画扇形并填充\*/

sector(500,300,180,270,200,100);/\*画椭圆扇形并填充\*/

delay(1000); /\*延时1秒\*/

}

cleardevice();

setcolor(14);

setfillpattern(str, RED);

bar(100,150,200,50);

bar3d(300,100,500,200,70,0);

pieslice(200,300,0,360,90);

sector(500,300,0,360,100,50);

getch();

getfillsettings(&save); /\*获得用户定义的填充模式信息\*/

closegraph();

clrscr();

printf("The pattern is %d, The color of filling is %d", save.pattern, save.color);

/\*输出目前填充图模和颜色值\*/

getch();

}

以上程序运行结束后, 在屏幕上显示出现行填充图模和颜色的常数值。

三、任意封闭图形的填充

截止目前为止, 我们只能对一些特定形状的封闭图形进行填充, 但还不能对

任意封闭图形进行填充。为此, TURBO C 提供了一个可对任意封闭图形填充的函

数, 其调用格式如下:

其中: x, y为封闭图形内的任意一点。border为边界的颜色, 也就是封闭图形轮廓的颜色。调用了该函数后, 将用规定的颜色和图模填满整个封闭图形。

注意:

1. 如果x或y取在边界上, 则不进行填充。

2. 如果不是封闭图形则填充会从没有封闭的地方溢出去, 填满其它地方。

3. 如果x或y在图形外面, 则填充封闭图形外的屏幕区域。

4. 由border指定的颜色值必须与图形轮廓的颜色值相同, 但填充色可选任意颜色。下例是有关floodfill()函数的用法, 该程序填充了bar3d()所画长方体中其它两个未填充的面。

例12:

#include<stdlib.h>

#include<graphics.h>

main()

{

int gdriver, gmode;

strct fillsettingstype save;

gdriver=DETECT;

initgraph(&gdriver, &gmode, "");

setbkcolor(BLUE);

cleardevice();

setcolor(LIGHTRED);

setlinestyle(0,0,3);

setfillstyle(1,14); /\*设置填充方式\*/

bar3d(100,200,400,350,200,1); /\*画长方体并填充\*/

floodfill(450,300,LIGHTRED); /\*填充长方体另外两个面\*/

floodfill(250,150, LIGHTRED);

rectangle (450,400,500,450); /\*画一矩形\*/

floodfill(470,420, LIGHTRED); /\*填充矩形\*/

getch();

closegraph();

}

6. 有关图形窗口和图形屏幕操作函数

一、图形窗口操作

象文本方式下可以设定屏幕窗口一样, 图形方式下也可以在屏幕上某一区域设定窗口, 只是设定的为图形窗口而已, 其后的有关图形操作都将以这个窗口的左上角(0,0)作为坐标原点, 而且可为通过设置使窗口之外的区域为不可接触。这样, 所有的图形操作就被限定在窗口内进行。

void far setviewport(int xl,int yl,int x2, int y2,int clipflag);

设定一个以(xl,yl)象元点为左上角, (x2,y2)象元为右下角的图形窗口, 其中x1,y1,x2,y2是相对于整个屏幕的坐标。若clipflag为非0, 则设定的图形以外部分不可接触, 若clipflag为0, 则图形窗口以外可以接触。

void far clearviewport(void);

清除现行图形窗口的内容。

void far getviewsettings(struct viewporttype far \* viewport);

获得关于现行窗口的信息,并将其存于viewporttype定义的结构变量viewport 中, 其中viewporttype的结构说明如下:

struct viewporttype{

int left, top, right, bottom;

int cliplag;

};

注明: 1. 窗口颜色的设置与前面讲过的屏幕颜色设置相同, 但屏幕背景色和窗口背景色只能是一种颜色, 如果窗口背景色改变, 整个屏幕的背景色也将改变这与文本窗口不同。

2. 可以在同一个屏幕上设置多个窗口, 但只能有一个现行窗口工作, 要对其它窗口操作, 通过将定义那个窗口的setviewport()函数再用一次即可。

3. 前面讲过图形屏幕操作的函数均适合于对窗口的操作。

二、屏幕操作

除了清屏函数以外, 关于屏幕操作还有以下函数:

void far setactivepage(int pagenum);

void far setvisualpage(int pagenum);

这两个函数只用于EGA,VGA 以及HERCULES图形适配器。setctivepage() 函数是为图形输出选择激活页。所谓激活页是指后续图形的输出被写到函数选定的pagenum页面, 该页面并不一定可见。setvisualpage()函数才使pagenum 所指定的页面变成可见页。页面从0开始(Turbo C默认页)。如果先用setactivepage() 函数在不同页面上画出一幅幅图像,再用setvisualpage()函数交替显示, 就可以实现一些动画的效果。

void far getimage(int xl,int yl, int x2,int y2, void far \*mapbuf);

void far putimge(int x,int,y,void \* mapbuf, int op);

unsined far imagesize(int xl,int yl,int x2,int y2);

这三个函数用于将屏幕上的图像复制到内存,然后再将内存中图像送回到屏幕上。首先通过函数imagesize()测试要保存左上角为(xl,yl), 右上角为(x2,y2)的图形屏幕区域内的全部内容需多少个字节, 然后再给mapbuf分配一个所测数字节内存空间的指针。通过调用getimage()函数就可将该区域内的图像保存在内存中, 需要时可用putimage()函数将该图像输出到左上角为点(x, y)位置上,其中getimage()函数中参数op规定如何释放内存中图像。关于这个参数的定义参见表8。

表8. putimage()函数中的op值

━━━━━━━━━━━━━━━━━━━━━━━━━━

符号常数 数值 含 义

──────────────────────────

COPY\_PUT 0 复制

XOR\_PUT 1 与屏幕图像异或的复制

OR\_PUT 2 与屏幕图像或后复制

AND\_PUT 3 与屏幕图像与后复制

NOT\_PUT 4 复制反像的图形

━━━━━━━━━━━━━━━━━━━━━━━━━━

对于imagesize()函数, 只能返回字节数小于64K字节的图像区域, 否则将会出错, 出错时返回-1。

本节介绍的函数在图像动画处理、菜单设计技巧中非常有用。

例13: 下面程序模拟两个小球动态碰撞过程。

#include<stdio.h>

#include<graphics.h>

int main()

{

int i, gdriver, gmode, size;

void \*buf;

gdriver=DETECT;

initgraph(&gdriver, &gmode, "");

setbkcolor(BLUE);

cleardevice();

setcolor(LIGHTRED);

setlinestyle(0,0,1);

setfillstyle(1, 10);

circle(100, 200, 30);

floodfill(100, 200, 12);

size=imagesize(69, 169, 131, 231);

buf=malloc(size);

getimage(69, 169, 131, 231,buf);

putimage(500, 269, buf, COPY\_PUT);

for(i=0; i<185; i++){

putimage(70+i, 170, buf, COPY\_PUT);

putimage(500-i, 170, buf, COPY\_PUT);

}

for(i=0;i<185; i++){

putimage(255-i, 170, buf, COPY\_PUT);

putimage(315+i, 170, buf, COPY\_PUT);

}

getch();

closegraph();

}

7. 图形模式下的文本输出

在图形模式下, 只能用标准输出函数, 如printf(), puts(), putchar() 函数输出文本到屏幕。除此之外, 其它输出函数(如窗口输出函数)不能使用, 即是可以输出的标准函数, 也只以前景色为白色, 按80列, 25行的文本方式输出。

Turbo C2.0也提供了一些专门用于在图形显示模式下的文本输出函数。下面将分别进行介绍。

一、文本输出函数

void far outtext(char far \*textstring);

该函数输出字符串指针textstring所指的文本在现行位置。

void far outtextxy(int x, int y, char far \*textstring);

该函数输出字符串指针textstring所指的文本在规定的(x, y)位置。 其中x 和y为象元坐标。

说明:

这两个函数都是输出字符串, 但经常会遇到输出数值或其它类型的数据, 此时就必须使用格式化输出函数sprintf()。

sprintf()函数的调用格式为: int sprintf(char \*str, char \*format, variable-list);

它与printf()函数不同之处是将按格式化规定的内容写入str 指向的字符串中, 返回值等于写入的字符个数。

例如: sprintf(s, "your TOEFL score is %d", mark);

这里s应是字符串指针或数组, mark为整型变量。

二、有关文本字体、字型和输出方式的设置

有关图形方式下的文本输出函数, 可以通过setcolor()函数设置输出文本的颜色。另外, 也可以改变文本字体大小以及选择是水平方向输出还是垂直方向输出。

void far settexjustify(int horiz, int vert);

该函数用于定位输出字符串。

对使用outtextxy(int x, int y, char far \*str textstring) 函数所输出的字符串, 其中哪个点对应于定位坐标(x, y)在Turbo C2.0中是有规定的。如果把一个字符串看成一个长方形的图形, 在水平方向显示时, 字符串长方形按垂直方向可分为顶部, 中部和底部三个位置, 水平方向可分为左, 中, 右三个位置,两者结合就有9个位置。

settextjustify()函数的第一个参数horiz指出水平方向三个位置中的一个,第二个参数vert指出垂直方向三个位置中的一个, 二者就确定了其中一个位置。 当规定了这个位置后, 用outtextxy()函数输出字符串时, 字符串长方形的这个规定位置就对准函数中的(x, y)位置。而对用outtext()函数输出字符串时, 这个规定的位置就位于现行游标的位置。有关参数horiz和vert的取值参见表9。

表9. 参数horiz和vert的取值

━━━━━━━━━━━━━━━━━━━━━━━━

符号常数 数值 用于

────────────────────────

LEFT\_TEXT 0 水平

RIGHT\_TEXT 2 水平

BOTTOM\_TEXT 0 垂直

TOP\_TEXT 2 垂直

CENTER\_TEXT 1 水平或垂直

━━━━━━━━━━━━━━━━━━━━━━━━

void far settextstyle(int font, int direction, int charsize);

该函数用来设置输出字符的字形(由font确定)、输出方向(由direction确定)和字符大小(由charsize确定)等特性。Turbo C2.0对函数中各个参数的规定见下列各表所示:

表10. font的取值

━━━━━━━━━━━━━━━━━━━━━━━━

符号常数 数值 含义

────────────────────────

DEFAULT\_FONT 0 8\*8点阵字(缺省值)

TRIPLEX\_FONT 1 三倍笔划字体

SMALL\_FONT 2 小号笔划字体

SANSSERIF\_FONT 3 无衬线笔划字体

GOTHIC\_FONT 4 黑体笔划字

━━━━━━━━━━━━━━━━━━━━━━━━

表11. direction的取值

━━━━━━━━━━━━━━━━━━━━━━━━

符号常数 数值 含义

────────────────────────

HORIZ\_DIR 0 从左到右

VERT\_DIR 1 从底到顶

━━━━━━━━━━━━━━━━━━━━━━━━

表12. charsize的取值

━━━━━━━━━━━━━━━━━━━━━━━━

符号常数或数值 含义

────────────────────────

1 8\*8点阵

2 16\*16点阵

3 24\*24点阵

4 32\*32点阵

5 40\*40点阵

6 48\*48点阵

7 56\*56点阵

8 64\*64点阵

9 72\*72点阵

10 80\*80点阵

USER\_CHAR\_SIZE=0 用户定义的字符大小

━━━━━━━━━━━━━━━━━━━━━━━━

有关图形屏幕下文本输出和字体字型设置函数的用法请看下例:

例14:

#include<graphics.h>

#include<stdio.h>

int main()

{

int i, gdriver, gmode;

char s[30];

gdriver=DETECT;

initgraph(&gdriver, &gmode, "");

setbkcolor(BLUE);

cleardevice();

setviewport(100, 100, 540, 380, 1); /\*定义一个图形窗口\*/

setfillstyle(1, 2); /\*绿色以实填充\*/

setcolor(YELLOW);

rectangle(0, 0, 439, 279);

floodfill(50, 50, 14);

setcolor(12);

settextstyle(1, 0, 8); /\*三重笔划字体, 水平放大8倍\*/

outtextxy(20, 20, "Good Better");

setcolor(15);

settextstyle(3, 0, 5); /\*无衬笔划字体, 水平放大5倍\*/

outtextxy(120, 120, "Good Better");

setcolor(14);

settextstyle(2, 0, 8);

i=620;

sprintf(s, "Your score is %d", i); /\*将数字转化为字符串\*/

outtextxy(30, 200, s); /\*指定位置输出字符串\*/

setcolor(1);

settextstyle(4, 0, 3);

outtextxy(70, 240, s);

getch();

closegraph();

return 0;

}

三、用户对文本字符大小的设置

前面介绍的settextstyle()函数, 可以设定图形方式下输出文本字符这字体和大小但对于笔划型字体(除8\*8点阵字以个的字体), 只能在水平和垂直方向以相同的放大倍数放大。为此Turbo C2.0又提供了另外一个setusercharsize() 函数, 对笔划字体可以分别设置水平和垂直方向的放大倍数。该函数的调用格式为:

void far setusercharsize(int mulx, int divx, int muly, int divy);

该函数用来设置笔划型字和放大系数, 它只有在settextstyle( ) 函数中的 charsize为0(或USER\_CHAR\_SIZE)时才起作用, 并且字体为函数settextstyle() 规定的字体。调用函数setusercharsize()后, 每个显示在屏幕上的字符都以其缺省大小乘以mulx/divx为输出字符宽, 乘以muly/divy为输出字符高。该函数的用法见下例。

例15:

#include<stdio.h>

#include<graphics.h>

int main()

{

int gdirver, gmode;

int gdriver=DETETC;

initgraph(&gdriver, &gmode, "");

setbkcolor(BLUE);

cleardevice();

setfillstyle(1, 2); /\*设置填充方式\*/

setcolor(WHITE); /\*设置白色作图\*/

rectangle(100, 100, 330, 380);

floodfill(50, 50, 14); /\*填充方框以外的区域\*/

setcolor(12); /\*作图色为淡红\*/

settextstyle(1, 0, 8);/\*三重笔划字体, 放大8倍\*/

outtextxy(120, 120, "Very Good");

setusercharsize(2, 1, 4, 1);/\*水平放大2倍, 垂直放大4倍\*/

setcolor(15);

settextstyle(3, 0, 5); /\*无衬字笔划, 放大5倍\*/

outtextxy(220, 220, "Very Good");

setusercharsize(4, 1, 1, 1);

settextstyle(3, 0, 0);

outtextxy(180, 320, "Good");

getch();

closegraph();

return 0;

}