·计算机应用·

文章编号:1006－7329(2001)01－0099－05

软件开发中的窗口滚动技术*

蓝 英， 吴维杰， 幸晓珂
(重庆大学B区 机电工程学院，重庆 400045)

摘要:提出了在DOS环境下，如何实现仿Windows中的窗口滚动技术，并编制出通用模块，可供直接移植调用，为软件开发者提供了极大的方便。

关键词:软件开发；窗口滚动技术；模块
中图分类号:TP311.52 **文献标识码**:A

在应用软件的研制开发中，窗口滚动是软件设计者所关注的重要技术之一，尤其是当有大量的数据或文字需要输出时，更希望数据窗口可以自由滚动。众所周知，Windows直观灵活的滚动窗口是极富吸引力的。要实现DOS下仿Windows的窗口滚动，需要熟悉屏幕滚动的中断调用，且需一定的编程构思和技巧。笔者应用Borland C++编制出在DOS下仿Windows的窗口滚动通用模块，可供软件开发者直接调用。

1 窗口滚动的技术实现

在开发的软件中，往往既有数据计算功能，又有图形显示功能，而图形和数据能够同时直观显示当然是最受用户欢迎的。要在有限的屏幕上同时显示图形和大量的数据时，就不得不借助于灵活的数据滚动视窗。窗口滚动的技术主要有以下两种：

1.1 移位重写法[1]

这种技术就是每次将需要输出的数据中的若干行写到屏幕上指定的窗口中以覆盖前次输出到窗口中的数据，当然每次只是根据上滚或下滚而更新窗口中最下或最上几行的数据，而其它各行的数据只是改变行位置后重新输出，给人以滚动的感觉，达到滚动的目的。

应用这种技术实现滚动，每次都要重写窗口中的所有数据，所以运行速度较慢，而且同一行中前后两次输出的数据所占列数不一样时，就不能完全覆盖，所以效果较差，只适用于滚动输出少量对称数据的情况。故在此就不做详细介绍。

1.2 屏幕滚动技术[2]

这种技术就是利用函数int86直接调用Bios中断，来实现屏幕的上滚和下滚。函数int86的用法如下：

其基本结构为：

```
#include <bios.h>
int86(int intr_num,union REGS *inregs,union REGS *outregs)
```

其中：*intr_num*——调用的Bios的中断号，实现屏幕滚动调用的中断号为10H

* 收稿日期:1999－09－14
作者简介:蓝 英(1977－)，女，四川广安人，硕士，主要从事机械设计及理论研究。

union REGS *inregs——定义的输入联合体

union REGS *outregs——定义的输出联合体

调用此函数时，首先定义联合体 union REGS r，然后对四个寄存器 AX(AH,AL)、BX(BH,BL)、CX(CH,CL)、DX(DH,DL)赋值。各值含义为：

AH=6H 时屏幕实现上滚，AH=7H 时屏幕实现下滚；

AL=屏幕滚动的行数(>0 上滚，<0 下滚，=0 窗口清屏)；

BH=屏幕滚动后留下的空白行的颜色值，BL=0；

CH——滚动窗口左上角坐标(行号)； CL——滚动窗口左上角坐标(列号)；

DH——滚动窗口右下角坐标(行号)； DL——滚动窗口右下角坐标(列号)。

根据以上的原理，编制窗口滚动的通用模块如下：

```
void scroll(m1,m2,n1,n2,lines,attrib)
char m1,m2,n1,n2,lines,attrib;
{union REGS r;
if(lines>0)
r.x.ax=0x0600+lines%256;
else
r.x.ax=0x0700+(abs(lines))%256;
r.x.bx=(attrib≪8)&0xff00;
r.x.cx=((m1-1)≪8)+n1-1;r.x.dx=((m2-1)≪8)+n2-1;
int86(0x10,&r,&r);
}
```

模块中参数的含义为：

m1——滚动窗口左上角坐标(行)； n1——滚动窗口左上角坐标(列)；

m2——滚动窗口右下角坐标(行)； n2——滚动窗口右下角坐标(列)；

lines——希望屏幕滚动的行数(>0 上滚，<0 下滚)；

attrib——希望屏幕滚动后留下的空白行的颜色。

以上的模块不论是在文本模式下，还是在图形模式下都可以通用。因此有了这样的通用模块后，调用它就可以实现窗口的任意行的滚动。当窗口中原有数据滚动后，会留下|lines|行空白行，这时再向这些空白行中输出新的数据，这就直观的实现了数据的滚动和更新。采用这种滚动技术，每次只需向空白行写入少量数据，大大提高了运行速度，因此非常适合于有大量数据或文字需要滚动输出的情况。

2 窗口滚动的控制及应用举例

在人机交互中，数据窗口的滚动可以通过键盘控制，也可以通过鼠标控制(这又涉及到鼠标通讯技术，限于篇幅，在此不做详述)。以下例举的为文本模式下键盘控制的程序，键盘对话的基本程序结构为：

```
do
{key=bioskey(0);
 if(key= =UPKEY) { scroll(m1,m2,n1,n2,lines,attrib);……;}
 if(key= =DOWNKEY) { scroll(m1,m2,n1,n2,lines,attrib);……;}
 ……;
 }while(key! = ESCKEY);
```

有了这样的结构后,当我们按下已定义的控制键时,一个栩栩滚动的窗口就会尽现屏幕之上。

以下为源程序:

```
#include <bios.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <stdlib.h>
#define UPKEY     0x4800
#define DOWNKEY 0x5000
#define ESCKEY    0X011b
#define ENTER     0X1c0d
#define MAX_M 6
void dis(char cf,char cb,char *cf1,char *cb1);
void scroll(char m1,char m2,char n1,char n2,char lines,char attrib);
void hk( char m,char n,char x,char y,char cf,char cb);
char cdback[400];
void main( )
  {char cf1,cb1,cf=7,cb=0,n=5;
  dis(cf,cb,&cf1,&cb1);cf=cf1;cb=cb1;
  getch(); textattr(0x07);clrscr();exit(0);
}
void dis(char cf,char cb,char *cf1,char *cb1)
  {int n1=22,m1=5,dn=4,ii=12,jj=32,i,j;
  int kk,lr,key;char c1,c,cm;char destin[1100];
  char *cofb[16]={"Black","Blue","Green","Cyan",
                  "Red","Magenta","Brown","White",
                  "Gray","BrBlue","BrGreen","Brcyan",
                  "BrRed","Pink","Yellow","BrWrite"};
  gettext(n1,m1,n1+jj+3,m1+ii+2,destin);
  gotoxy(n1+5,m1+2);cprintf("Foregronnd");hk(9,8,n1+dn,m1+3,0,7);
  if(cf<8){c1=7;cm=cf;} else{c1=cf;cm=7;}
  for(i=0;i<=7;i++){gotoxy(n1+dn+1,m1+i+4);cprintf("%s",cofb[c1-7+i]);}
  textattr(0x0f);gotoxy(n1+dn+1,m1+cm+4);cprintf("%s",cofb[cf]);
  lr=0;*cf1=cf;*cb1=cb;
do
{key=bioskey(0);
 if(key==ESCKEY){cf=*cf1;cb=*cb1;goto QQQ;}
  if(key==DOWNKEY)
  {if(lr==0&&cf<15)
{if(cf<7)
{textattr(0x70);gotoxy(n1+dn+1,m1+cf+4);cprintf("%s",cofb[cf]);
 cf++;textattr(0x0f);gotoxy(n1+dn+1,m1+cf+4);cprintf("%s",cofb[cf]);
```

```
 c=16*cb+cf;
}
else
{textattr(0x70);cf++;gotoxy(n1+dn+1,m1+11);cprintf("%s",cofb[cf-1]);
scroll(9,16,27,35,1,7);
textattr(0x0f);gotoxy(n1+dn+1,m1+11);cprintf("%s",cofb[cf]);
}
c=16*cb+cf;
}
if(lr==1&&cb<7)
   {textattr(0x70);gotoxy(n1+dn+16,m1+cb+4);cprintf("%s",cofb[cb]);cb++;
   textattr(0x0f);gotoxy(n1+dn+16,m1+cb+4);cprintf("%s",cofb[cb]);c=16*cb+cf;
}
}
   if(key==UPKEY)
   {if(lr==0&&cf>0)
    {if(cf<8)
{textattr(0x70);gotoxy(n1+dn+1,m1+cf+4);cprintf("%s",cofb[cf]);cf--;
 textattr(0x0f);gotoxy(n1+dn+1,m1+cf+4);cprintf("%s",cofb[cf]);
}
   else
{scroll(9,16,27,35,-1,7); cf--;textattr(0x0f);
 gotoxy(n1+dn+1,m1+11);cprintf("%s",cofb[cf]);
 textattr(0x70);gotoxy(n1+dn+1,m1+4);cprintf("%s",cofb[cf-7]);
}
c=16*cb+cf;
}
if(lr==1&&cb>0)
{textattr(0x70);gotoxy(n1+dn+16,m1+cb+4);cprintf("%s",cofb[cb]);cb--;
textattr(0x0f);gotoxy(n1+dn+16,m1+cb+4);cprintf("%s",cofb[cb]);c=16*cb+cf;
    }
   }
} while(key!=ENTER);
QQQ:
   puttext(n1,m1,n1+jj+3,m1+ii+2,destin);
   if(!(cb==*cb1&&cf==*cf1))
   {for(j=1;j<=23;j++)
    for(i=1;i<=159;i+=2)
    pokeb(0xb800,160*j+i,16*cb+cf); *cf1=cf; *cb1=cb;
    }
   }
void scroll(char m1,char m2,char n1,char n2,char lines,char attrib)
{union REGS r;
```

```
 if(lines>0) r.x.ax=0x0600+lines%256;
 else
  r.x.ax=0x0700+(abs(lines))%256;r.x.bx=(attrib<<8)&0xff00;
  r.x.cx=((m1-1)<<8)+n1-1;r.x.dx=((m2-1)<<8)+n2-1;
  int86(0x10,&r,&r);
}
void hk( char m,char n,char x,char y,char cf,char cb)
{unsigned char j,c,k,kg[80];
 kg[0]=218;kg[m+1]=191;kg[m+2]='\0';
 for(k=1;k<=m;k++) kg[k]=196;
 c=16*cb+cf;textattr(c);gotoxy(x,y);cprintf("%s",kg);
 for(k=1;k<=n;k++)
 {gotoxy(x,y+k);cprintf("%c",179);
  for(j=1;j<=m;j++)
  {textattr(c);gotoxy(x+j,y+k);cprintf("%c",32);}
  textattr(c); gotoxy(x+m+1,y+k);cprintf("%c",179);}
  kg[0]=192;kg[m+1]=217;gotoxy(x,y+n+1);cprintf("%s",kg);}
```

参考文献:

[1] 谭浩强. C 语言程序设计[M]. 北京:高等教育出版社,1996
[2] 施小龙. Borland C++深入编程[M]. 北京:学苑出版社,1994

Study on Window Rolling Technology in Software Development

LAN Ying, *WU Wei-jie*, *XING Xiao-ke*
(Faculty of Mechanical and Electrical Engineering, Chongqing University B, Chongqing 400045, China)

Abstract: In this paper, the ways of simulating the window rolling technology in DOS environment were described. A general module, which is easy to be transplanted and useful for users, is presented.
Keywords: software development; window rolling technology; module