УДК 519.725

ОБ ОДНОМ НЕСТАНДАРТНОМ ФОРМАТЕ ДЛЯ ХРАНЕНИЯ ГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ

© Р.В. Кольцов

Koltsov R.V. The Effective Format of Grafical Images Storage. The article contains notes about effective format for keeping graphic images, for IBM PC compatible computers, which can be successfully used in shareware. Besides the article contains listings of procedures in assembler of keeping images in a file and displaying them on the screen.

Правильный выбор формата для сохранения файлов графики обеспечивает компактное хранение изображений и позволяет экономить дисковое пространство. Иногда выставляются критичные требования к объемам файлов. Такая проблема часто возникает при создании больших проектов, включающих быструю графику, либо при разработке игровых программ и т.п. Свой выбор, конечно, можно остановить на каком-либо стандартном формате, например .PCX, .GIF, .TIF, .PIC [1] и .BMP [2], которые включают в себя помимо собственно кода различную инициализирующую информацию. Но что делать, если Вы пишите приложение для какого-то конкретного графического режима и вся лишняя информация Вам не нужна, да и объем критичен? Возможно, у Вас возникнет желание создать нестандартный файл, который посторонний пользователь не сможет просмотреть? Либо же вы желаете программировать один из недокументированных режимов графики? [3, 4].

Здесь рассматривается нестандартный формат для хранения образа экрана 4-плоскостного графического режима 10h (640×350 точек). Эта статья написана как альтернатива части главы 7 [3], где приводятся листинги сохранения и восстановления экранов в режиме 10h без использования сжатия данных, которые весьма неудобны для применения. Здесь Вашему вниманию представлен формат, использующий простой, но достаточно эффективный алгоритм сжатия. В файле образа хранятся подряд байты блоков данных. Каждому блоку предшествует свой байт-счетчик `Count`. Никакая дополнительная информация, типа массивов палитры, в файле не хранится. При желании Вы можете разместить ее непосредственно в своей программе. При записи файла образа каждый блок данных формируется в буфере `Buffer` вместе с байтом `Count`, которые по мере заполнения сбрасываются в файл. Байт `Count` имеет следующую структуру: если его бит 7 равен нулю, то младшие 6 бит показывают число неупакованных байтов в последующем блоке, а если бит 7 равен единице, то биты 0-6 показывают, сколько в последующем байте, являющим собою целый блок, упаковано байтов из видеобуфера. Очевидно, что таких байтов, упаковывающихся в один, не может быть больше 127. При записи изображения в файл данные последовательно считываются из четырех битовых плоскостей видеобуфера, причем по столбцам сверху вниз и слева направо для достижения максимального сжатия. Практика показывает, что чаще лучшее сжатие достигается по столбцам, а не по строкам. Это связано с организацией видеобуфера ПЭВМ. Не трудно приведенный алгоритм адаптировать для других видеорежимов, таких как 0Dh, 0Eh, 0Fh, 12h и др. Ключевые места в листингах снабжены комментариями. Поменяв VGA_SEGMENT с 0A000h на 0A6D6h, Вы сможете читать изображение из файла в невидимую область видеобуфера, с последующим копированием в его видимую часть. Тестируемое графическое изображение в формате .SWE заняло объем 47943 байт, а в формате .PCX 78775 байт.

Листинг 1.1. Данные к процедурам

```
VGA_SEGMENT equ 0A000h
GC_INDEX equ 3CEh ;Индексный регистр
                  ;GC - графического контроллера
DISPLAYED_SCREEN_SIZE     equ     (640/8)*350
LineBuf db 28000 dup (0)
Filename db 'SCREEN.SWE',0
Handle dw ? ;Дескриптор файла
FileLen dw ?
;Count  и  нижеследующий  буфер  вместе
;сбрасываются  по  мере  заполнения  в  файл
Count db 0
Buffer db 127 dup (0)
;Флаг  =  1,  если  требуется  сброс  (update)
UEnableFlag db 0
```

Листинг 1.2.

```
;Процедура записи изображения из видеобуфера
;в открытый методом дескриптора файл File
SaveTheScreen PROC near
  push ds es di si
  cld
  mov dx,GC_INDEX
  mov ax,0005h ;индекс регистра режима GC
  out dx,ax
  mov ax,0304h
```

```
  mov cx,4
10:push cx
  out dx,ax
  push ax
  mov ax,0A000h
  mov es,ax
;В LineBuf считываем по очереди 4 плоскости
  mov si,offset LineBuf
  xor di,di
  mov cx,80
111:push cx
  mov cx,350
112:mov bl,es:[di]
  mov [si],bl
  add di,80
  inc si
  loop 112
  sub di,DISPLAYED_SCREEN_SIZE-1
  pop cx
  loop 111
;Изображение из одной плоскости уже в
;LineBuf.Ниже мы обрабатываем по очереди
;считанные плоскости, сжимая данные,
;заполняем сжатыми данными Buffer и Count,
;которые по мере заполнения сбрасываем на
;диск.
  push ds
  pop es
  mov si,offset LineBuf
  mov di,offset LineBuf
InitCount:mov Count,0
  lodsb
  mov cl,al
  push si
  lodsb
  cmp al,cl
  jnz b13
  lodsb
  cmp al,cl
  jnz b13
  mov Count,80h
b13:pop si
;Далее: al - байт, считанный предпоследним
;(LAST),cl - байт, считанный последним (CUR)
13:push cx
  lodsb
  mov cl,al
  pop ax ;LAST:=CUR, CUR:=ds:[si],si++
  cmp si,(offset LineBuf+2)+ \
          DISPLAYED_SCREEN_SIZE
  jc fw ;LineBuf исчерпан? Переход,если нет
  ja QuitLoop
  stosb
  inc Count
  jmp QuitLoop ;исчерпан
fw:cmp al,cl ;LAST = CUR ?
  jnz m3 ;на "левую ветвь" алгоритма
  push ax ;сохраним LAST
  lodsb ;посмотрим NEXT BYTE в LineBuf
  dec si ;не меняя там своей позиции
  cmp al,cl ;равен ли он CUR ?
  pop ax ;восстановим LAST в al
  jz met1 ;перейти, если NEXT BYTE = CUR
mm1:cmp Count,80h ;Count > 80
  jb m3 ;нет
  cmp Count,255
  jz fwd1
mm2:inc Count
  cmp Count,255
  jnz fwd0
  mov al,cl
  jmp fwd1
fwd0:inc Count
  call Update
  jmp InitCount
m3:cmp Count,127 ;Если Buffer заполнен
  jnz m4          ;неповторяющимися байтами,
fwd1:call Update ;то сбросить его в файл
  mov Count,0 ;Инициализируем верно счетчик
m4:mov UEnableFlag,1
  stosb        ;Это операция LAST -> Buffer
  inc Count
  jmp 13
met1:cmp Count,80h ;"Правая ветвь" алгоритма
  jc fwd2
met2:cmp Count,255
  jnz met3
fwd2:call Update
  mov Count,80h
met3:mov UEnableFlag,1
  stosb
  inc Count
  jmp 13
QuitLoop:cmp UEnableFlag,0
  jz EOPlane
  call Update
EOPlane:pop ax
  dec ah
  pop cx
  loop j10
  pop si di es ds
  ret
j10:jmp 10
SaveTheScreen ENDP
;Процедура, сбрасывающая в файл счетчик чис-
;ла байтов Count и Count байтов из Buffer'a
Update PROC near
  push ax bx cx dx
  mov bx,Handle ;дескриптор
;ds:dx -> буфер, откуда записываем в файл
  mov dx,offset Count
  mov ch,0
  mov cl,Count ;cx = число записываемых байт
  test cl,80h
  jz $+4
  mov cl,1
  inc cx ;$+4
  mov ah,40h;Dos Fn "WriteToFile via Handle"
  int 21h
  mov UEnableFlag,0
  mov di,offset Buffer
  pop dx cx bx ax
  ret
Update ENDP
;Процедура прорисовки битовых плоскостей
FillPlane PROC near
  out dx,ax
  xor di,di
  mov ah,01h
  mov cx,80
11:push cx
  mov cx,350
12:dec ah
  test ah,7Fh
  jz m1
  test ah,80h
  jnz m2
  lodsb
  jmp m2
m1:lodsb
  mov ah,al
  lodsb
m2:mov bl,es:[di]
  stosb
  add di,79
  loop 12
  sub di,DISPLAYED_SCREEN_SIZE-1
  pop cx
  loop 11
  ret
FillPlane ENDP
;Процедура заполнения видеобуфера из буфера
;Buf.На входе ax=сегмент видеобуфера
RestoreTheScreen PROC near
  push ds ;Для видимой части 0A000h
  cld     ;Для невидимой части 0A6D0h
  mov dx,03CEh
  mov es,ax
  call ClearVolumeVisibleScreen
;Здесь Buf - это буфер, куда Вы прочитаете
свой графический файл. Переименуйте его,
если он у Вас имеет другое имя
  mov ds,Buf
  xor si,si
  mov ax,1003h ;Операция "OR"
  out dx,ax
  mov ax,0701h ;пишем в плоскость 3
  call FillPlane
  mov ax,0B01h ;пишем в плоскость 2
  call FillPlane
  mov ax,0D01h ;пишем в плоскость 1
  call FillPlane
  mov ax,0E01h ;пишем в плоскость 0
  call FillPlane
  mov ax,0105h
  mov dx,03CEh
  out dx,ax
  pop ds
```

```
  ret
RestoreTheScreen ENDP
;Процедура очистки всего экрана
ClearVolumeVisibleScreen PROC near
  mov ax,0005h
  out dx,ax
  mov ax,0FF08h ;Модифицируем все пикселы
  out dx,ax
  mov ax,0003h
  out dx,ax
  mov ax,0000h
  out dx,ax
  mov ax,0F01h
  out dx,ax
  xor di,di
  mov cx,DISPLAYED_SCREEN_SIZE/2
  rep stosw     ;ax -> es:[di]
  ret
ClearVolumeVisibleScreen ENDP
;Процедура непосредственно показывающая
;графическое изображение на экране и
;ожидающая нажатия любой клавиши
;На входе ds:dx -> Filename
ShowPicture PROC near
  call FillBufFromFile
  mov ax,VGA_SEGMENT
  call RestoreTheScreen
  mov ah,08h
  int 21h
  ret
ShowPicture ENDP
;Процедура заполнения буфера Buf
;из файла SCREEN.SWE
;На входе ds:dx -> Filename
FillBufFromFile PROC near
  call OpenFile
  mov Handle,ax
  call ReadFile
  mov bx,Handle
  call CloseFile
  ret
FillBufFromFile
;Процедура открытия файла. В случае успеха
;возвращает в ax дескриптор файла
OpenFile PROC near
  mov ah,3Dh
  mov al,02h
  int 21h
  jc Not_open
  ret
Not_open:mov ax,0003h
  int 10h
  mov dx,offset OpenErrMessage
  mov ah,09h
  int 21h
  mov ax,4CFFh ;Errorlevel = 0FFh - Error!
  int 21h
OpenFile ENDP
;Процедура чтения файла целиком в Buf
ReadFile PROC near
  mov ax,4202h
  xor cx,cx
  mov dx,cx
  mov bx,Handle
  int 21h
  mov FileLen,ax
  mov ah,4200h
  xor dx,dx
  int 21h
  mov bx,Handle
  mov cx,FileLen
  xor dx,dx
  mov ah,3Fh
  push ds
  mov ds,Buf
  int 21h
  pop ds
  ret
ReadFile ENDP
;Процедура закрытия файла
;На входе в bx дескриптор закрываемого файла
CloseFile PROC near
  mov ah,3Eh
  int 21h
  ret
CloseFile ENDP
```

ЛИТЕРАТУРА

1. *Романов В.Ю.* Популярные форматы файлов для хранения графических изображений на IBM PC. М.: Унитех, 1992. 160 с.
2. *Сван Том.* Windows. Форматы файлов. М.: Бином, 1995. 288 с.
3. *Абраш Майкл.* Программирование графики. Таинства. Киев: EvroSYB, 1995. 384 с.
4. *Шикин Е.В., Боресков А.В.* Компьютерная графика. Динамика, реалистические изображения. М.: Диалог-МИФИ, 1995. 228 с.

Поступила в редакцию 5 июня 1996 г.