Ingo Eickmann

Panning

Große Text- und Grafikseiten verwalten

EGA und VGA können Textseiten und Grafiken speichern, die über ihr Auflösungsvermögen weit hinausgehen. Der Bildschirm zeigt dann einen frei wählbaren Ausschnitt, dessen Größe dem aktuellen Videomodus entspricht. Wir stellen die Programmierung des panning vor.

CAD-Programme haben den Begriff des *panning* in der Computerei populär gemacht. Mit panning wird ein Anzeigemodus bezeichnet, bei dem nur ein Teil einer Gesamtansicht angezeigt wird und verschoben werden kann. Die Bewegung des dargestellten Ausschnitts erfolgt meist durch Eingabe eines Aufpunktes und eines Verschiebungsvektors. Die Anzeige kann dadurch über der gesamten Ansicht frei bewegt werden, was unser Aufmacher veranschaulichen soll.

Die Software macht's

Das panning bei CAD-Programmen für Personalcomputer erfolgt meist softwaremäßig. Der Bildschirmspeicher wird mit der veränderten Ansicht neu geladen. Die universelle Programmierbarkeit des *Cathode-Ray-Tube-Controllers* (CRTC) und des *Attribute-Controllers* einer EGA oder VGA lassen ein Hardware-panning zu. Hierbei befindet sich das gesamte Bild ständig im Videospeicher. Die Controller werden so programmiert, daß sie lediglich den gewünschten Ausschnitt des Video-RAM zur Anzeige bringen. Die Bildschirmanzeige erfolgt in einer Standardauflösung - beispielsweise 640x480 Pixel, wogegen das komplette Bild eine größere Auflösung besitzt, zum Beispiel 720x720 Bildpunkte. Die Vorteile des hardware panning sind seine Schnelligkeit und sein geringer Speicherplatzbedarf. Da der angezeigte Bildausschnitt nicht im Speicher bewegt wird, sondern lediglich die Controller bei einer Verschiebung anders auf den Videospeicher zugreifen, kann das angezeigte Bild unabhängig von der Zahl der dargestellten Bildpunkte sehr schnell bewegt werden. Soll der sichtbare Bildteil Pixel für Pixel verschoben werden, ein Vorgang, den man als *smooth panning* bezeichnet, begrenzt lediglich die Bildfrequenz die Geschwindigkeit der Bewegung. Bei einer Bildfrequenz von 60Hz im 640x480 Modus kann das Bild um maximal 60 Pixel pro Sekunde verschoben werden, da jede Verschiebung um einen Pixel einmal zur Anzeige gelangen muß. Selbstverständlich kann der Bildschirminhalt auch schneller verschoben werden, indem das Bild jeweils gleich um mehrere Pixel versetzt wird. Auch die sofortige Anzeige eines anderen Ausschnitts ist möglich. Die Geschwindigkeit des neuen Bildaufbaus bleibt unabhängig von der Auflösung.

Hardware panning belegt im Vergleich zu Software panning keinen zusätzlichen Speicher. Beim hardware panning befindet sich das gesamte Bild ständig im Videospeicher. Beim Software panning muß das Bild komplett im Hauptspeicher liegen, und der angezeigte Teil des Bildes wird in den sichtbaren Teil des Videospeichers bewegt. In den Text- und Grafikmodi wird der CRT-Controller in ähnlicher Weise programmiert, um aus einem großen Bild im Videospeicher einen Bildausschnitt zu zeigen. Die Belegung des Videospeichers wird gegenüber den Standardmodi prinzipiell beibehalten. Alle Zeilen des gesamten Bildes werden nacheinander im Videospeicher abgelegt. Da die Zeilen eines größeren Bildes jedoch breiter werden, ändert sich die Speicheraufteilung gegenüber dem ursprünglichen Videomodus. Der CRT-Controller, der die Daten zum Aufbau der Anzeige aus dem Videospeicher holt, wird mit der zur linken oberen Ecke des Bildausschnitts gehörenden Speicherstelle programmiert. Hierzu wird deren Offset in das Startadreßregisterpaar des Controllers geschrieben. Diese Möglichkeit bieten alle Videoadapter mit dem CRT-Controller 6845. Daher wäre bereits bei einer CGA- oder Hercules-Grafikkarte zeilenweises Scrollen möglich gewesen, was aber wegen des zu geringen Videospeichers nicht genutzt wurde.

Da bei der Programmierung erweiterter Auflösungen die Bildzeilen gegenüber der Speicherverwaltung im ursprünglichen Videomodus breiter geworden sind, muß dem CRT-Controller die neue Breite über Offset-Register, das erst seit der EGA-Karte zur Verfügung steht, mitgeteilt werden. Das Beschreiben der CRTC-Register erfolgt synchron zum Katodenstrahl. Details über die Programmierung des CRT-Controllers finden Sie

in der nächsten mc. Bei den Textmodi kommt zur pixelweisen Verschiebung des Bildausschnitts noch die Information hinzu, bei welcher Pixelposition innerhalb der Zeichenmatrix der Bildausschnitt in der linken oberen Ecke beginnen soll. Die Nummer der Rasterzeile wird hierzu in das *Preset-Row-Scan-Register* des CRTC geschrieben. Die Position des Ursprungs in dieser Rasterzeile innerhalb der Zeichenmatrix wird im *Horizontal-Pixel-Panning-Register* des Attribute-Controllers abgelegt.

Assemblermodule

Bild 1 enthält alle Assemblermodule, um große Text- und Grafikbilder im Videospeicher zu verwalten und um das panning durchzuführen. Die Prozeduren *Expand-Text* und *PanText* initialisieren eine erweiterte Textseite und legen, den sichtbaren Bildschirmausschnitt fest. *Bild 2* zeigt die Einbindung der Assemblermodule als Unit in Turbo Pascal. Das Programm *pandemo.pas* in *Bild 3* verwendet die Assemblermodule, um eine 132x100 Zeichen große Textseite zu öffnen und mit einem Testbild zu beschreiben. Der Bildschirm zeigt einen 80x25 Zeichen großen Ausschnitt im ursprünglich angewählten Videomodus 3 (Text 80x25 Color). Dieser Ausschnitt ist im gesamten Bereich der 132x100 Zeichen pixelweise frei verschiebbar. Die Programme sind für die neunspaltige Zeichenmatrix des VGA-Textmodus ausgelegt. In den Kommentaren sind die Änderungen für die achtspaltige Zeichenmatrix des EGA-Textmodus vermerkt. Bei der Initialisierung einer größeren Textseite mit *ExpandText(PX,PY)* ist unbedingt darauf zu achten, daß die Pixelbreiten *PX* und *PY* Vielfache der Zeichenmatrixhöhe oder -breite sind. Die neue Textseite muß größer als die bisherige (80x25) sein und darf nicht mehr Speicherplatz erfordern, als ein Segment zur Verfügung stellt, also 64 KByte. Hierbei müssen wir uns immer vor Augen halten, daß der CRT-Controller lediglich in einem festen Segment den Videospeicher mittels Offset adressieren kann. Mit den Funktionen *ExpandGraphic* und *PanGraphic* aus Bild 1 erzeugt das Demoprogramm *pandemo.pas* ein 720x720 Pixel großes Testbild im VGA-Farbgrafikmodus. Der angezeigte Bildbereich mit der Standardauflösung von 640x480 Bildpunkten kann über dem gesamten Bild frei bewegt werden.
Achtung Turbo-Pascal-Freunde: Aufgrund der neuen Belegung des Videospeichers lassen sich die pixelbearbeitenden Befehle der Unit Graph nicht mehr verwenden. Als Beispiel wurde die neue Version des *PutPixel*-Befehls für erweiterte Grafikseiten implementiert. Er heißt nun *ExpandPutPixel* und wird mit den gleichen Parametern aufgerufen, wie der ursprüngliche Befehl, vergleiche dazu Bild 2. Erst das Umschreiben des *Borland Graphic Interface* (BGI) ermöglicht die Verwendung der ursprünglichen Befehle der Unit *Graph.* Die Assemblermodule lassen sich auch für die übrigen Text- und Grafikmodi nutzen. Lediglich beim MCGA-Grafikmodus 320x200 in 256 Farben ist Vorsicht geboten. Hier repräsentiert ein Byte ein Pixel und nicht mehr 8 Pixel einer Map.

Textseiten

Prinzipiell lassen sich große Textseiten, wie beispielsweise 132x100 Zeichen unter DOS nutzen. Breite Assemblerlistings können dadurch auch mit einer normalen EGA oder VGA betrachtet werden. Damit DOS die neue Aufteilung des Videospeichers berücksichtigen kann, müssen Statusinformationen im Datenbereich des Video-BIOS an die neuen Spalten- und Zeilenzahlen angepaßt werden. Die *Tabelle* zeigt alle Informationen über den Videostatus im BIOS-Bereich, die durch Aufrufe des Interrupt 10h im Systemsegment 0040h verwaltet werden. Bei den Offsets 4Ah, 4Ch und 84h müssen wir neue Werte für die gesamte Auflösung im erweiterten Textmodus einsetzen. Diese Änderungen führt das Assemblermodul *ExpandText* automatisch durch. Leider reagiert der *CLS*-Befehl nicht auf eine geänderte Zeilenzahl. Er löscht weiterhin nur 25 Zeilen. Darüberhinaus holt DOS beim Scrollen die Farbe für die unterste Zeile aus dem nachfolgenden Eintrag des Videospeichers. Damit die unterste Zeile nach dem Scrollen ein bestimmtes attribut byte erhält, muß dieses also bei der Initialisierung unmittelbar nach der erweiterten Textseite im Videospeicher folgen. Standardsoftware und DOS arbeiten meist nicht problemlos mit neuen Auflösungen zusammen. Das gilt sowohl für die hier beschriebene Programmierung einer erweiterten Auflösung, als auch für die neuen Text- und Grafikmodi von Super-EGAs und Super-VGAs. Grafische Oberflächen verweigern die Zusammenarbeit mit einer erweiterten Anzeige, sofern sie nicht auf beliebige Spalten- und Zeilenzahlen umkonfigurierbar sind. Die Anwendung neuer Text- und Grafikauflösungen im Zusammenspiel mit panning hat gezeigt, daß sie bei der Benutzung von Standardsoftware nicht zweckmäßig ist, jedoch für selbstgeschriebene Programme völlig neue Auflösungen und Scroll-Möglichkeiten erschließt.

Literatur

[1] *Wilton, R.:* The programmer's guide to PC and PS/2 Video Systems.
Microsoft Press, 1987.

Videostatus im BIOS-Bereich		
Offset	**Inhalt**	**Type**
49h	Nummer des Videomodus	Byte
4Ah	Anzahl der angezeigten Zeichen pro Zeile	Word
4Ch	Größe des Videospeichers in Bytes	Word
4Eh	Offset der Startadresse im Videospeicher	Word
50h	Zeiger auf Cursor-Position Array (8 Worte): Für jede Bildschirmseite ein Wort, bestehend aus Textzeile (MSB) und Textspalte (LSB)	Word
60h	Cursorblock, bestehend aus oberster Rasterzeile (MSB) und unterster Rasterzeile (LSB)	Word
62h	Nummer der angezeigten Textseite	Byte
63h	Port des CRTC-Adreßregisters (3B4h Monochrom, 3D4h Farbe)	Word
65h	Inhalt des Mode-Control-Registers (3B8h bei monochromen Modes, 3D8h bei Color Modes), von EGA und VGA emuliert	Byte
66h	Inhalt des CGA Color-Select-Registers (3D9h), von EGA und VGA emuliert	Byte
84h	Anzahl der angezeigten Textzeilen - 1	Byte
85h	Höhe der Character Matrix in Rasterzeilen	Word
87h	EGA/VGA-Info-Byte 1: Bit 7: Bit 7 des zuletzt mit INT 10 initialisierten Videomodus Bit 6,5: 0: 64 KByte RAM 1: 128 KByte RAM 2: 192 KByte RAM 3: 256 KByte RAM Bit 4: (reserviert) Bit 3: Video-Subsystem ist nicht aktiv Bit 2: (reserviert) Bit 1: Video-Subsystem ist an ein monochromes Display angeschlossen Bit 0: Emulation des alphanumerischen Cursors	Byte
88h	EGA/VGA-Info-Byte 3: Inputs vom Feature Connector: Bit 7: Bit 6 des Input-Status-Registers 0 als Antwort auf Bit 1 des Feature-Control-Registers Bit 6: Bit 5 des Input-Status-Registers 0 als Antwort auf Bit 1 des Feature-Control-Registers Bit 5: Bit 6 des Input-Status-Registers 0 als Antwort auf Bit 0 des Feature-Control-Registers Bit 4: Bit 5 des Input-Status-Registers 0 als Antwort auf Bit 0 des Feature-Control-Registers EGA-Konfigurationsschalter (von VGA emuliert): Bit 3: Schalter 4 off Bit 2: Schalter 3 off Bit 1: Schalter 2 off Bit 0: Schalter 1 off	Byte
89h	MCGA/VGA-miscellaneous flags: Bit 7,4: 0: 350 Rasterzeilen pro Seite 1: 400 Rasterzeilen pro Seite 2: 200 Rasterzeilen pro Seite 3: (reserviert) Bit 6: Display switching enabled Bit 5: (reserviert) Bit 3: Default palette loading disable Bit 2: Monochromer Monitor angeschlossen Bit 1: Gray scale summing enabled Bit 0: VGA aktiv geschaltet	Byte
8Ah	Index für display combination code table	Byte
A8h	Vektor auf save pointer table	DWord

Das Systemsegment 0040h enthält die Statusinformationen über das aktuelle Display, die vom Video BIOS über Interrupt 10h verwaltet werden.

```
;------------------------------------------------------------------------------
;     Panning für Turbo Pascal 4.0/5.x und EGA/VGA                release 1.0
;     Copyright (c) Ingo Eickmann, 1989                             11/05/89
;------------------------------------------------------------------------------

PAGE 65,132
TITLE Panning

   BufferLength    equ  4Ch           ; Eintrag im Systemsegment (s. Tabelle)
   Columns         equ  4Ah           ; Eintrag im Systemsegment (s. Tabelle)
   CRTC            equ  63h           ; Eintrag im Systemsegment (s. Tabelle)
   EGABase         equ  0A000h        ; Segment des Grafikvideospeichers
   GraphicColNorm  equ  640           ; alte Auflösung (X) des Grafikmode 12h
   GraphicRowNorm  equ  480           ; alte Auflösung (Y) des Grafikmode 12h
   PixelPerChar    equ  9             ; für VGA, HGC und EGA mono Mode,
                                      ; 8 für alle anderen EGA Modes
   Rows_m1         equ  84h           ; Eintrag im Systemsegment (s. Tabelle)
   SLinesPerChar   equ  85h           ; Eintrag im Systemsegment (s. Tabelle)
   TextColNorm     equ  80            ; alte Auflösung (X) des Textmode 3
   TextRowNorm     equ  25            ; alte Auflösung (Y) des Textmode 3

if1
   VertRet macro n                    ; Vertical Retrace (Bit 3) im
           local Loco                 ; Input Status Register 1 abfragen und
           Loco: in al,dx             ; auf  0 (nz: No Vertical Retrace)
                 test al,00001000b    ; bzw. 1 ( z: Vertical Retrace)
                 j&n Loco             ; warten.
           endm
;------------------------------------------------------------------------------
SetPanning macro                      ; Programmierung der CRTC und Attribute
              mov bl,al               ; Controller Register
              mov dx,es:[CRTC]
              add dl,6                ; ermitteln des CRTC Status Registers
              VertRet nz              ; auf das Ende eines Vert.Retrace warten
              VertRet z               ; auf den Anfang eines Horz. Ret. warten
              cli
              sub dl,6                ; CRTC Address Register
              mov al,12
              out dx,ax               ; Start Address (High) belegen
              inc al
              mov ah,bl
              out dx,ax               ; Start Address (Low) belegen
              mov ah,ch
              mov al,8
              out dx,ax               ; Preset Row Scan Register belegen
              mov dl,0C0h             ; Horizontal Pixel Panning Register
              mov al,33h              ; im Attribute Controller adressieren
              out dx,al
              mov al,bh
              out dx,al               ; Horz. Pixel Panning Register belegen
              sti
              mov al,1                ; Return Value für function:Boolean:=.t.
           endm
;------------------------------------------------------------------------------
MExpandText macro PX,PY,ETRaus        ; Erweiterung der Auflösung
              xor al,al               ; im Textmode 3
              mov bx,TextColNorm
              cmp bx,PX               ; Auflösung in X-Richtung zulässig?
              jge ETRaus              ;   NEIN ==> Raus
              mov bx,TextRowNorm
              cmp bx,PY               ; Auflösung in Y-Richtung zulässig?
              jge ETRaus              ;   NEIN ==> Raus
              mov ax,PX
              mov bx,PY
              mul bx                  ; Anzahl der Character pro erweiterter
              mov cx,ax               ; Bildschirmseite
              dec ax
              and ax,8000h
              or  dx,ax               ; Anzahl der Character > 8000h ?
              jnz ETRaus              ;   JA ==> Raus
              dec bx
              or  bh,bh               ; Anzahl der Textzeilen > 256 ?
              jnz ETRaus              ;   JA ==> Raus
```

```
             mov es:[BufferLength],cx ; neue Größe der Bildschirmseite
             mov ax,PX                ; speichern (s. Tabelle)
             mov es:[Columns],ax      ; Anzahl der Spalten und (Zeilen - 1)
             mov es:[Rows_m1],bl      ; im Systemsegment ablegen
             shr ax,1                 ; Worte pro Textzeile
             mov ah,al
             mov al,19
             mov dx,es:[CRTC]         ; Offset Register des CRTC neu
             out dx,ax                ; programmieren
             mov al,1                 ; Return Value für function:Boolean:=.t.
           endm
;-------------------------------------------------------------------------
MPanText macro Px,Py,PTRaus           ; angezeigtes Window durch die absoluten
        local noPT,Not9,Tokay         ; Pixelkoordinaten der linken oberen
             mov ax,PX                ; Ecke festlegen
             xor dx,dx
             mov cx,PixelPerChar      ; X-Pixelkoordinate in Columns umrechnen
             div cx
             mov bh,dl                ; bh: Pixel Pan Value
             mov bl,al                ; bl: Char Offset in Row
             dec bh                   ; ************************
             jge Not9                 ; *** Entfällt für EGA ***
             mov bh,8                 ; ************************
        Not9: add ax,TextColNorm
             cmp ax,es:[Columns]      ; rechter Rand der Bildschirmseite durch
             jg  noPT                 ; Window überschritten? - JA ==> Raus
             mov ax,PY
             xor dx,dx
             mov cx,es:[SLinesPerChar]
             div cx                   ; Y-Pixelkoordinate in Rows umrechnen
             or  dh,ah                ; Anzahl der Textzeilen zulässig?
             jz  Tokay                ;   JA ==> Okay
        noPT: xor al,al               ; Return Value für function:Boolean:=.f.
             jmp PTRaus               ; Raussprung
       Tokay: mov ch,dl               ; ch: Preset Scan Line
             mov cl,al                ; cl: Row Offset
             add ax,TextRowNorm-1
             cmp ax,es:[Rows_m1]      ; unterer Rand der Bildschirmseite durch
             jg  noPT                 ; Window überschritten? - JA ==> Raus
             mov al,cl                ; Startadresse berechnen:
             xor ah,ah
             mul word ptr es:[Columns] ;         Zeilenzahl * Zeilenlänge
             xor dx,dx                ;    +          Offset in der Zeile
             mov dl,bl                ;   ------------------------------
             add ax,dx                ;   Startadresse für den CRTC in AX

             SetPanning               ; Programmierung des CRTC und
         endm                         ; des Attribute Controllers
;-------------------------------------------------------------------------
MExpandGraphic macro PX,PY,EGRaus     ; Erweiterung der Auflösung im
                xor al,al             ; Grafikmode 12h
                mov bx,GraphicColNorm
                cmp bx,PX             ; Auflösung in X-Richtung zulässig?
                jg  EGRaus            ;   NEIN ==> Raus
                mov bx,PY
                cmp bx,GraphicRowNorm ; Auflösung in Y-Richtung zulässig?
                jl  EGRaus            ;   NEIN -> Raus
                mov ax,PX             ; Berechnung des Speicherbedarfs für
                mov cl,4              ; die angeforderte Auflösung
                shr ax,cl             ; durch 8 teilen und nach rechts shiften
                mul bx
                dec ax
                and ax,8000h
                or  dx,ax             ; Speicherbedarf > 65536 Bytes / Map ?
                jnz EGRaus            ;   JA ==> Raus
                mov ax,PX
                mov xGMax,ax          ; neue Auflösung im Datensegment ablegen
                mov yGMax,bx
                shr ax,cl             ; Words pro Pixelzeile
                mov ah,al
                mov al,19
                mov dx,es:[CRTC]      ; Offset Register des CRTC neu
                out dx,ax             ; programmieren
```

```
                mov al,1              ; Return Value für function:Boolean:=.t.
              endm
;-------------------------------------------------------------------------
MPanGraphic macro PX,PY,PGRaus        ; angezeigtes Window durch die absoluten
           local noPG,Gokay           ; Pixelkoordinaten der linken oberen
                mov ax,PX             ; Ecke festlegen
                mov bx,ax
                add bx,GraphicColNorm
                cmp bx,xGMax          ; rechter Rand der Bildschirmseite durch
                jg  noPG              ; Window überschritten? - JA ==> Raus
                xor dx,dx
                mov cx,8              ; 8 Pixel / Byte bei Grafikmodus 12h
                div cx
                mov bh,dl             ; Pixel Pan Value
                mov bl,al             ; Char Offset in Row
                mov ax,PY
                mov cx,ax
                add cx,GraphicRowNorm
                cmp cx,yGMax          ; unterer Rand der Bildschirmseite durch
                jle Gokay             ; Window überschritten? - No ==> Okay
           noPG: xor al,al            ; Return Value für function:Boolean:=.f.
                jmp PGRaus            ; Raussprung
          Gokay: mov dx,xGMax         ; Startadresse berechnen:
                mov cl,3              ;   Zeilenzahl * Zeilenlänge in Bytes
                shr dx,cl
                mov cx,dx
                mul cx
                xor ch,ch             ; ch:=0 für Addition und Preset Row Scan
                mov cl,bl             ; cx: Row Offset
                add ax,cx             ; Offset in der Zeile addieren
                SetPanning            ; Programmierung des CRTC und AttribCtrl
            endm
endif

data segment word public
          xGMax dw 0                  ; neue Grafikauflösung für ExpandGraphic
          yGMax dw 0
data ends

code segment word public 'CODE'
     assume cs:code,ds:data

IRP fname,<ExpandText,PanText,ExpandGraphic,PanGraphic>
   ;-----------------------------------------------------------------------
   ; function &fname(PX,PY: Integer):Boolean;
   ;-----------------------------------------------------------------------
   public &fname
   &fname proc far

   PY equ [bp+6]                      ; die Parameter liegen in umgekehrter
   PX equ [bp+8]                      ; Reihenfolge auf dem Stack
            push bp                   ; Kopf der Funktionsaufrufe
            mov  bp,sp
            push es
            mov  ax,40h               ; Segmentregister ES auf
            mov  es,ax                ; Systemsegment setzen
            mov  al,ah                ; function:Boolean:=.f. setzen
            mov  bx,PX
            or   bx,PY                ; ist ein Parameter negativ?
            jge  MJP&fname            ;   NEIN ==> Macro
            jmp  Raus&fname           ;   JA   ==> Raus
MJP&fname:  M&fname PX,PY,Raus&fname  ; Makro der Funktion
Raus&fname: pop  es                   ; Raussprung
            pop  bp
            ret  4                    ; 2 Intergerwerte belegten 4 Bytes
   &fname endp                        ; zusätzlich auf dem Stack
endm

;-------------------------------------------------------------------------
; procedure ExpandPutPixel(PX,PY,Color: Integer);
;-------------------------------------------------------------------------
public ExpandPutPixel                 ; Pixel im erweiterten Grafikmodus
ExpandPutPixel proc far               ; setzen
```

```
   Color equ [bp+6]
   PY    equ [bp+8]                  ; die Parameter liegen in umgekehrter
   PX    equ [bp+10]                 ; Reihenfolge auf dem Stack
              push bp                ; Kopf der Procedure
              mov  bp,sp
              push es
              mov  dx,3CEh           ; Adresse des Graphic Controllers
              mov  ax,Color          ; Die Pixelfarbe ..
              xor  ah,ah
              xchg ah,al             ; .. selektiert im Set-/Reset-Register
              out  dx,ax             ;    das Beschreiben der Maps
              mov  ax,0F01h          ; Alle 4 Set-/Reset-Register werden
              out  dx,ax             ; zum Beschreiben der Maps zugelassen
              mov  ax,3              ; Data Rotate Register mit Rot. Count 0
              out  dx,ax             ; und keine Farbmodifikation belegen
              mov  al,8              ; Bit Mask Register vorab anwählen
              out  dx,al
              mov  cx,PX             ; Ist X-Koordinate innerhalb des
              mov  ax,xGMax          ; virtuellen Bildes?
              cmp  ax,cx
              jle  EPRaus            ;   NEIN ==> Raus
              and  cl,7              ; Bit Maske ermitteln
              mov  al,10000000b
              shr  al,cl
              inc  dx                ; und im Bit Mask Register ablegen
              out  dx,al
              mov  bx,PY             ; Ist Y-Koordinate innerhalb des
              mov  cx,yGMax          ; virtuellen Bild?
              cmp  bx,cx
              jge  EPRaus            ;   NEIN ==> Raus
              mov  ax,xGMax          ; Adresse im Videospeicher berechnen:
              mov  cl,3
              shr  ax,cl             ;   Y * X-Auflösung / 8 Pixel/Byte
              push dx
              mul  bx
              pop  dx
              mov  bx,PX             ;              +  X / 8 Pixel/Byte
              shr  bx,cl
              add  ax,bx
              mov  bp,ax
              mov  ax,EGABase        ; Segment des Videospeichers im
              mov  es,ax             ; Grafikmode 12h
              mov  al,es:[bp]        ; Lese- und Schreiboperation mit
              mov  es:[bp],al        ; dieser Speicherstelle ausführen
     EPRaus:  mov  al,0FFh           ; Alle Bits der im Bit Mask Register
              out  dx,al             ; wieder zulassen
              dec  dx
              mov  ax,1              ; Set-/Reset-Register zurücksetzen
              out  dx,ax
              pop  es                ; verwendete Register wieder herstellen
              pop  bp
              ret  6                 ; 3 Integer-Parameter belegten 6 Bytes
ExpandPutPixel endp                  ; zusätzlich auf dem Stack

code ends                            ; MS-C: Segmentname code in _TEXT ändern
     end
```

Bild 1. panning.asm enthält alle Assemblermodule zur Initialisierung und Verwaltung von Grafik- und Textfenstern

```
unit Panning;

(* Unit - Declaration für Turbo Pascal 4.0/5.x, I.Eickmann 1989 *)

{$F+}

interface
  function  ExpandText(PX,PY : Integer) : boolean;
  function  PanText(PX,PY : Integer) : boolean;
  function  ExpandGraphic(PX,PY : Integer) : boolean;
  function  PanGraphic(PX,PY : Integer) : boolean;
  procedure ExpandPutPixel(PX,PY,Color : Integer);

implementation
  {$L Panning.obj}
  function  ExpandText;     external;
  function  PanText;        external;
  function  ExpandGraphic;  external;
  function  PanGraphic;     external;
  procedure ExpandPutPixel; external;

end.
```

Bild 2. Die Unit panning.pas bindet die Assemblermodule ein und stellt sie dem Pascal-Programmierer zur Verfügung

```
{------------------------------------------------------------------------}
{     Panning Demo for VGA                                    release 1.0 }
{     Copyright (c) Ingo Eickmann, 1989                          11/07/89 }
{------------------------------------------------------------------------}
{     Compilieren mit Turbo Pascal 4.0/5.x               I. Eickmann      }
{                                                        Im Leuchterbruch 8 }
{     Getestet unter MS-DOS 3.3                          5000 Köln 80     }
{------------------------------------------------------------------------}
{     Je ein Testbild für den Textmode 80x25 und den Grafikmode 640x480    }
{     wird angezeigt. Die erweiterte Textseite umfaßt 132x100 Zeichen,     }
{     die erweiterte Grafikseite 720x720 Pixel.                            }
{     Das angezeigte Window hat die Größe des original Modes und kann mit  }
{     den Pfeiltasten frei in der erweiterten Seite bewegt werden.         }
{     HOME/PageUp und END/PageDn springen zum Anfang bzw. Ende der         }
{     erweiterten Seite, ESC bricht ab.                                    }
{------------------------------------------------------------------------}

program PanDemo;

uses CRT,DOS,Panning;

const xTextMax  = 132; xGraphicMax  = 720;        { Auflösung der          }
      yTextMax  = 100; yGraphicMax  = 720;        { neuen Modi             }
      xTextNorm =  80; xGraphicNorm = 640;        { Auflösung der          }
      yTextNorm =  25; yGraphicNorm = 480;        { original Modes         }

      PixelPerChar = 9;  (* für VGA, HGC und EGA mono Modes,
                            8 für alle anderen EGA Textmodes *)

var   Line,Column,SLinesPerChar           : Integer;
      LineDir,LineMax,ColumnDir,ColumnMax : Integer;
      Taste1,Taste2                       : Char;
      Regs                                : Registers;
      Text,Error                          : Boolean;

procedure MoveWindow;                    { Der Bildschirm zeigt einen frei  }
                                         { verschiebbaren Ausschnitt der    }
var Error : Boolean;                     { erweiterten Bildschirmseite      }

begin
  Line:=0;                               { Windowkoordinaten initialisieren }
  Column:=0; Error:=false;
  repeat                                 { Hauptschleife beginnt hier       }
    Taste1:=chr(0); Taste2:=chr(0);
```

```
    if keypressed then                 { Tastatur abfragen                 }
    begin
      Taste1:=ReadKey;
       if (Taste1=chr(0)) and keypressed then Taste2:=ReadKey;
    end;
    if Taste1<=chr(0) then
    begin
      case Taste2 of
{ PgUp } chr(73),chr(71) : begin
{ Home }                     Line:=0;                      { Tastatur-      }
                             Column:=0;                    { eingaben       }
                             LineDir:=0;                   { verarbeiten    }
                           end;
{ PgDn } chr(81),chr(79) : begin
{ End }                      Line:=LineMax;
                             Column:=0;
                             LineDir:=0;
                           end;
{ Left }  chr(75) :        ColumnDir:=ColumnDir - 1;
{ Right } chr(77) :        ColumnDir:=ColumnDir + 1;
{ Up }    chr(72) :        LineDir:=LineDir - 1;
{ Down }  chr(80) :        LineDir:=LineDir + 1;
      end;
    end;

    if Taste1=' ' then
    begin
      LineDir:=0;                                          { Stop panning   }
      ColumnDir:=0;
    end;
    Line:=Line + LineDir;
    if Line<0 then
    begin
      Line:=0;                                             { oberer Rand ?  }
      LineDir:=0;
    end;
    if Line>LineMax then
    begin
      Line:=LineMax;                                       { unterer Rand ? }
      LineDir:=0;
    end;
    Column:=Column + ColumnDir;
    if Column<0 then
    begin
      Column:=0;                                           { linker Rand ?  }
      ColumnDir:=0;
    end;
    if Column>ColumnMax then
    begin
      Column:=ColumnMax;                                   { rechter Rand ? }
      ColumnDir:=0;
    end;
    if Text then Error:= not PanText(Column,Line)
            else Error:= not PanGraphic(Column,Line);
    if Error then writeln('Out of frame: Line ', Line, ' Column ', Column);
  until Taste1=chr(27);                    { Hauptschleife beenden mit ESC }
end;

Begin                                                  { Hauptroutine      }
  SLinesPerChar:=MemW[$0040:$0085];
  LineDir:=0;
  ColumnDir:=0;
  Text:=true;
  LineMax:=(yTextMax-yTextNorm) * SLinesPerChar;
  ColumnMax:=(xTextMax-xTextNorm) * PixelPerChar;
  if ExpandText(xTextMax,yTextMax) then
  begin
    for Line:=0 to pred(yTextMax) do                   { Testbild darstellen   }
    begin
      for Column:=0 to pred(xTextMax) do
```

```
      begin
        if (Column mod 10)=0
        then
          MemW[$B800:$0+(Line*xTextMax+Column) shl 1]:=$7160+trunc(Column/10)
        else if (Column mod 10)=5
             then MemW[$B800:$0+(Line*xTextMax+Column) shl 1]:=$7430+Line
             else MemW[$B800:$0+(Line*xTextMax+Column) shl 1]:=$0730+Column
mod 10;
      end;
    end;
    MoveWindow;
  end
  else writeln('Error ! - No TextWindow opened.');
  Text:=false;
  with Regs do
  begin
    AX:=$12;                                        { In den Grafikmode 12h }
    Intr($10,Regs);                                 { umschalten            }
  end;
  LineMax:=yGraphicMax-yGraphicNorm;
  ColumnMax:=xGraphicMax-xGraphicNorm;
  if ExpandGraphic(xGraphicMax,yGraphicMax) then
  begin
    Error:=false;                                   { Testbild darstellen   }
    for Line:=0 to pred(yGraphicMax) do
    begin
      ExpandPutPixel(0,Line,white);
      ExpandPutPixel(pred(xGraphicMax),Line,white);
      ExpandPutPixel(trunc((1-Line/yGraphicMax)*xGraphicMax),Line,red);
      ExpandPutPixel(trunc(Line/yGraphicMax*xGraphicMax),Line,red);
    end;
    for Column:=0 to pred(xGraphicMax) do
    begin
      ExpandPutPixel(Column,0,white);
      ExpandPutPixel(Column,pred(yGraphicMax),white);
    end;
    MoveWindow;
  end
  else Error:=true;
  with Regs do
  begin
    AX:=3;                                          { In den Textmode       }
    Intr($10,Regs);                                 { zurückschalten        }
  end;
  if Error then writeln('Error ! - No GraphicWindow opened.');
End.
```

Bild 3. pandemo.pas initialisiert je ein großes Test- und Grafiktestbild.
Die Bildanzeige kann als Fenster frei über dem Testbild bewegt werden.