Turbo Pascal deel 3

MSX CLUB MAGAZINE 36

Erik van Bilsen

Scanned, ocr'ed and converted to PDF by HansO, 2001

In deze aflevering van de cursus gaan we scrollen en wel smooth-scroll in maar liefst vier richtingen. Moeilijk ? Met de bijgeleverde routines niet meer!

Foutje

In het bestand GRAPH1.LIB van de vorige aflevering is een foutje geslopen. Geëxperimenteer met bijvoorbeeld de kopieer-procedures heeft bij menigeen wellicht geleid tot het vastlopen van het programma. De fout zit in de absolute variabele ExpTbl. Onder de variabelen-declaratie staat de volgende regel:

```
ExpTbl: INTEGER ABSOLUTE $FCCO;
```

Deze regel moet worden vervangen door

```
ExpTbl:   INTEGER ABSOLUTE   $FCC1;
```

Het bovenstaande probleem is nu verholpen.

GRAPH2.LIB

Deze keer vindt u als bijlage op het diskabonnement het bestand GRAPH2.LIB. Dit bestand bevat een aantal extra variabelen en de routines Scroll, InitVerScroll en InitHorScrolL Dit bestand is evenals het bestand GRAPH1.UB ook slechts een aanzet tot het opzetten van een procedure-bibliotheek. Sommige procedures (met name de scroll-procedure) kunnen namelijk nog sneller door ze te programmeren in assembleertaaL Wilt u in uw programma gebruik maken van de smooth-scroll mogelijkheden, neem dan de volgende regels op na de variabele-dedaratie (de bestanden MSXBIOS.UB en GRAPH1.LB waren een bijlage bij de vorige aflevering):

```
{$I MsxBios.lib}
{$I Graph1.lib}
{$I Graph2.1ib}
```

Ik ga in deze aflevering niet te diep in op de technische specificaties van de video-chip. Hiervoor verwijs ik naar de cursus "Ken uw computer: de V9938 videochip" van Emil Hensen uit MSX dub Magazine nummers 25-27.

Verticale scrolls

Verticaal scrollen is in principe heel eenvoudig. Het wordt bewerkstelligd door VDP-regjster 23 (in BASIC VDP(24)) telkens met l te verhogen of te verlagen. Dit kan zelfs in BASIC op de volgende manier.

```
10  SCREEN x
20 'Kies voor x scherm-mode 5, 6, 7 of 8
30 BLOAD "plaatje.pic",S
40 'Laadt een willekeurige tekening in
50 VDP(24)=(VDP(24)+1) AND 255
60 GOTO 50
```

Het probleem is dan dat telkens dezelfde pagina wordt gescrolld. Dit is voor de meeste toepassingen niet de bedoeling. Zoals je ziet wordt ook het in eerste instantie onzichtbare video-ram (het video-ram waarin de gegevens over de sprites en kleuren worden opgeslagen) meegescrolld. Dit houdt in dat de sprite-gegevens ergens anders moeten worden opgeslagen; willen we het volledige scherm kunnen gebruiken. Hiertoe moet voorafgaand aan het scrollen de procedure InitVerScroll worden aangeroepen Deze procedurebeschrijving vindt u op de volgende pagina.
Voor het verticaal scrollen kunnen de schermmodes 5 t/m 8 worden gebruikt Afhankelijk van de ingestelde schermmode wordt het onzichtbare videoram gewist, met de FillVram procedure. Vervolgens wordt m.b.v. de SetPage procedure de actieve pagina gezet op de pagina waarop de karakters staan. Voor schermmodes 5 en 6 is dit page 2 en voor de schermmodes 7 en 8 page 1. Zorg dus vooraf dat in deze pagina's bovenaan de karakters staan. Door de waarden van vdp-registers 6 en 11 te veranderen worden de sprite- en kleur-gegevens op een andere pagina gezet. Tot slot wordt de variabele ScrolIPage geladen met het paginanummer waar de karakters staan.
Elke keer als de procedure Scroll wordt aangeroepen wordt het scherm één pixel in de aangegeven richting gescrolld. De richting is een variabele van het type Direc-tions (enumeratie), en kan de waarden up, down, left of right hebben. De procedure kan bijvoorbeeld als volgt worden aangeroepen:

Scroll(down);

De scroll-procedure, voor zover die betrekking heeft op het verticaal scrollen, is hiernaast weergegeven.

Gebruik procedure

Elke keer als het scherm 8 pixels is gescrolld moet er een nieuwe regel karakters worden afgedrukt. De routines zijn geschreven voor de schermmodes 5 en 8, die beide 32 karakters breed zijn. De karakternummers worden opgeslagen in
de variabele ScrollTable, een array van 32 bytes groot. De variabele ScrollTable bevat dus de karakternummers van de nieuw af te drukken regel. Twee mogelijkheden zijn er nu. De eerste luidt

- Scroll het scherm 8 pixels;

- Laadt de variabele ScrollTable met 32 nieuwe karakters;
- Druk die karakters af;
- Herhaal het voorgaande.

Deze methode leidt ertoe dat het scrollen niet soepel verloopt, omdat na elke scroÜ van 8 pixels even gewacht moet worden om de nieuwe karakters af te drukken. Vandaar dat voor een andere oplossing is gekozen. Deze houdt in dat iedere keer als het scherm één pixel wordt gescrolld, 4 karakters worden afgedrukt.

Voor deze tweede mogelijkheid is de werkwijze als volgt

- Vul de eerste vier array-ele-menten van de variabele ScrollTable met hun nieuwe waarde;
- Scroll het scherm l pixel;
- Vul de volgende vier array-elementen met hun nieuwe waarde;
- Scroll het scherm l pixel;
- Etc.int

In TP ziet het bovenstaande er bijvoorbeeld uit zoals uitgewerkt als in het kader hiernaast

Terug naar de procedure Scroll. Afhankelijk van de gekozen richting roept de procedure Scroll de sub-procedure VerScroll of HorScroll aan. Ik zal deze procedures verder wegens plaatsgebrek niet uitleggen. Een uitgebreid voorbeeld waarin van verticale scrolls wordt gebruik gemaakt (het programma VERSCROL.PAS) is opgenomen aan het eind van de aflevering en staat ook op het diskabonnement

Schuif-operatoren

Het valt op dat in de procedure Scroll diverse malen van de operatoren shl en shr gebruik gemaakt wordt. Dit zijn zogenaamde schuif-operatoren en ze zijn ook in assem-bleertaal bekend. Ze worden gebruikt om op een snelle manier getallen te vermenigvuldigen of te delen. Als we in het decimale stel-sen een getal naar links verschuiven, wordt het met 10 vermenigvuldigd (123.456 wordt 1234.56). Een verschuiving naar rechts betekent een deling door 10 (123.456 wordt 12.3456). De operatoren shl en shr zijn afkortingen van SHift Left en SHift Right, en hebben betrekking op het binaire talstelsel. Bijvoorbeeld:

```
VarNaam  shl   l
```

schuift alle bits in de variabele VarNaam l plaats naar links. Dit komt overeen met een vermenig vuldiging met 2. Evenzo is VarNaam shl 2 gelijk aan een vermenigvuldiging met 4, VarNaam shl 3 gelijk aan een vermenigvuldiging met 8 enzovoorts.
Shl en shr kunnen alleen worden gebruikt op variabelen van het type BYTE of INTEGER. Het gebruik van deze operatoren gaat tot 3 maal sneller dan het gebruik van de operatoren * en div (de deelope-rator / kan alleen op variabelen van het type REAL worden toegepast. Het delen van gehele getallen gaat m.b.v. de operator div).

Commentaar

In TP-programmatekst is op twee manieren commentaar of andere niet-programmatekst op te nemen. Commentaar kan geplaatst worden tussen twee accolades {commentaar } of tussen twee haakjes met sterretjes (* commentaar *).

Horizontale scrolls

Horizontaal scrollen is ingewikkelder. Er moet gebruik gemaakt worden van de SetAdjust-procedure om het gehele scherm l pixel naar links of naar rechts te verschuiven. Ik zal de procedure hier niet uitleggen, omdat zij ingewikkeld is. en zal volstaan met aan te geven hoe de procedure gebruikt moet worden.
Alvorens horizontaal te gaan scrollen moet eerst de procedure Init-HorScroll worden aangeroepen. Deze procedure initialiseert enkele variabelen. Vervolgens kan horizontaal worden gescrolld m.b.v. Scroll(left) of Scroll(right). De sub-procedure HorScroll scrollt de eerste 24 regels van het scherm en kan alleen worden gebruikt in de schermmodes 5 of 6. Normaal zijn deze schermmodes 26,5 regels hoog. De karakternummers van de nieuw af te drukken kolom moeten dit maal worden opgeslagen in de variabele HorScrollTable, een twee dimensionale array van 24 * 2 bytes. Omdat het scherm wordt gescrolld in delen van 16 pixels breed, moeten na elke scroll
van 16 pixels 2 nieuwe kolommen worden afgedrukt van 24 karakters hoog. De totale array is 24 * 2 = 48 bytes groot en moet in 16 keer worden afgedrukt Dit houdt in dat elke keer als de procedure Scroll wordt aangeroepen, drie nieuwe karakters worden afgedrukt.

Ga dus als volg te werk
- Vul de array-elementen [0,0], [0,1] en [1,0] van de variabele HorScrollTable met hun nieuwe waarde;
- Scroll l pixel;
- Vul de array-elementen [1,1], [2,0] en [2,1] met hun nieuw waarde;
- Scroll l pixel;
- Etc.

De procedure wisselt page 0 en page l voortdurend af. In page 2 staan de karakters. Een voorbeeld waarin gebruik wordt gemaakt van horizontale scrolls is opgenomen aan het eind van deze aflevering en staat ook op het diskabon-nement. Laadt van te voren in BASIC een willekeurig screen 5 plaatje in op page 2.
Zoals je ziet werkt de horizontale scroll procedure nog wat schokkerig. Dit is te verhelpen door binnen Turbo-Pascal assembleertaal-procedures te schrijven. Hoe dit moet behandel ik in een van de volgende afleveringen. De volgende aflevering ga ik in op het produceren van geluid en muziek met Turbo-Pascal.

```
PROGRAM HORSCROL;

{$I MsxBios.lib}
{$I Graph1.lib}
{$I Graph2.lib}

VAR
   Ytel,Xtel,Teller1,Teller2:BYTE;

BEGIN
   Color(15,0,0); Screen(5); SetPage(0,2); RestorePalet; InitHorScroll;
   REPEAT
      Ytel:=0; Xtel:=0;
      FOR Teller1:=0 TO 15 DO
         BEGIN
            FOR Teller2:=0 TO 2 DO
               BEGIN
                  HorScrollTable[Ytel,Xtel]:=Random(256);
                  Xtel:=Xtel+1;
                  IF Xtel=2 THEN
                     BEGIN
                        Xtel:=0;
                        Ytel:=Ytel+1
                     END;
               END;
            Scroll(right)
         END;
   UNTIL KeyPressed;
   Screen(0); NewPalet; Color(1,14,14); Cls
END.
```

```
PROGRAM VERSCROL;

{ Dit programma is een TP-versie van mijn inzending SCROLL.BAS
  in MSX CLUB magazine 28. Het smooth-scrolled door het gehele
  veld van BREAKER van Radarsoft. Ga als volgt te werk:
   - run het programma BREAKER
   - reset de computer
   - compileer en run dit programma

  Na het resetten van BREAKER bevat page 1 van screen 8 alle
  informatie over de opbouw van het veld:
   - Het bovenste deel bestaat uit de karakters die in het
     spel te zien zijn (8 * 8 pixels groot);
   - Het middelse deel (vanaf adres $4000) bestaat uit de
     opbouw van de blokken. Elk blok wordt vertegenwoordigd
     door 8 * 8 pixels, waarvan elke pixel het karakternummer
     aangeeft. Elk blok is dus 8 * 8 karakters groot;
   - Het onderste deel (tot adres $9000) bevat de opbouw van
     het veld. Op het scherm passen 4 blokken naast elkaar.
     In groepjes van vier bytes (pixels) wordt aangegeven
     welke blokken moeten worden afgedrukt. }

{$I MsxBios.lib}
{$I Graph1.lib}
{$I Graph2.lib}

VAR
   Teller1,Teller2,TabelTeller,Yblock,BlockNr: BYTE;
   ScreenTable,BlockTable,CharAdress: INTEGER;

BEGIN
   Color(15,0,0); Screen(8); InitVerScroll;
   Yblock:=7; ScreenTable:=$9000; BlockTable:=$4000;
   REPEAT           {Deze lus wordt herhaald totdat een toets wordt
ingedrukt}
      TabelTeller:=0;
      IF Yblock=7 THEN ScreenTable:=ScreenTable-4;
      FOR Teller1:=0 TO 3 DO                          {4 blokken naast
elkaar}
         BEGIN
            BlockNr:=Vpeek(ScreenTable+Teller1);
            CharAdress:=BlockTable+(BlockNr and  31) shl 3
                                  +(BlockNr and 224) shl 6
                                  +Yblock shl 8;
            FOR Teller2:=0 TO 7 DO          {8 karakters naast elkaar
in blok}
               BEGIN
                  ScrollTable[TabelTeller]:=Vpeek(CharAdress+Teller2);
                  TabelTeller:=TabelTeller+1;
                  IF Teller2=3 THEN Scroll(down)
               END;
            Scroll(down);
         END;
      Yblock:=(Yblock-1) and 7
   UNTIL KeyPressed;
   Screen(0); Color(1,14,14); Cls
END.
```