17.1 De video-adapter

Als er één ding sterk verbeterd is aan de PC, dan is het wel de weergave op het beeldscherm. Een beeldscherm is opgebouwd uit beeldpunten. Hoe kleiner het aantal beeldpunten is, des te grover het beeld gerasterd is. Bij de eerste systemen was het beeldscherm erg grof. Met name bij kleurenweergave leidde dit tot erg grove beelden, terwijl bovendien het aantal beschikbare kleuren zeer beperkt was. Inmiddels zijn er videosystemen ontwikkeld die tienduizenden kleurnuances kunnen weergeven. Een grafische afbeelding in zo'n systeem kan kwalitatief wedijveren met een kleurenfoto.

De manier waarop de video-adapter en de computer samenwerken, is in de loop der jaren niet zo erg veranderd. Er zijn in het geheugen twee segmenten gereserveerd voor het video-systeem: het A- en het B-segment. In dit deel van het geheugen bevindt zich de videobuffer. Onze programma's geven de machine opdracht om bepaalde gegevens in de buffer te plaatsen. De video-adapter tast zo'n vijftig tot zeventig keer per seconde de buffer af en stuurt de gegevens naar het beeldscherm.

Video-adapters kunnen op verschillende manieren ingesteld worden. Als we deze instellingen willen kennen, moeten we twee hoofdinstellingen onderscheiden: tekst-mode en grafische mode. Met grafische mode wordt dan het weergeven van tekeningen of foto's bedoeld. Er is maar één video-adapter die deze twee hoofdinstellingen niet kent, en dat is de Monochrome Display Adapter (MDA). Deze adapter zat in de eerste IBM PC en kende uitluitend de tekst-mode.

17.2 Tekstinstellingen

Voor alle adapters wordt tekst op dezelfde manier in het geheugen opgeslagen. Voor ieder teken dat op het scherm kan worden opgeslagen, hebben we twee bytes in het geheugen nodig. In de ene byte staat welk teken weergegeven moet worden, en in de andere hoe het teken moet worden afgebeeld. Als we een scherm gebruiken dat horizontaal 80 tekens kan afbeelden en waarop 25 regels gebruikt kunnen worden, dan hebben we voor het afbeelden van deze tekens maximaal 25 x 80 = 2000 bytes nodig. Deze bytes noemen we tekenbytes. Daarnaast hebben we 2000 bytes nodig om aan te geven hoe het teken moet worden afgebeeld. Voor één tekstscherm kunnen dus 4000 bytes nodig zijn. Deze bytes heten attribuutbytes.

Door deze manier van opslag correspondeert een adres in de videobuffer met een plaats op het scherm. Met de opdracht

GotoXY(40,12) sturen we de cursor naar de veertigste positie van de twaalfde regel. Stel dat we met een Write-opdracht de letter "A" willen afbeelden op de positie waar de cursor zich bevindt. We kunnen dan uitrekenen in welke bytes van de videobuffer de gegevens zullen komen te staan. Als we de gegevens kwijt moeten op de twaalfde regel, dan moeten we eerst elf regels overslaan. Dit zijn: 11 x 160 = 1760 bytes. Hier komen de 39 tekens van de twaalfde regel bij. Dat zijn 78 bytes. De gegevens van de letter worden dan geplaatst in 1760+78+1 = de 1839ste byte. In de 1840ste byte komt dan de voor- en achtergrondkleur te staan.

Ook de videobuffer heeft een adres in het geheugen. Een tekst-mode in kleur begint altijd op het adres hex B800:0000. Bij monochrome weergave daarentegen staat de videobuffer op het adres hex B000.0000. In Turbo Pascal worden adressen en interrupts voorafgegaan door een $-teken.

Nu weten we wel hoe de gegevens opgeslagen zijn; maar hoe werkt het nou? Om dit duidelijk te maken, splitsen we de attribuutbyte in tweeën. We hebben nu dus twee groepjes van vier bytes, oftewel twee nibbles. In hoofdstuk twee, dat werken met het scherm behandelt, staat een lijstje van beschikbare kleuren. De nummering van deze kleuren loopt van 0 tot en met 15. Deze nummering is niet willekeurig, maar afhankelijk van de aanwezigheid van de kleuren rood, groen of blauw in zijn kleurenspectrum. Uit combinaties van deze kleuren kunnen de kleuren vervaardigd worden die de computer op het scherm brengt.

Om aan te kunnen geven of rood, groen of blauw aangezet moeten worden, hebben we drie bits nodig. Blauw gebruikt bit 0, groen bit 1 en rood staat in bit 2. We hebben dan nog een bit over in de nibble. Dit is de helderheidsbit. Als deze bit aan staat, krijgen we lichte heldere kleuren, als hij uit staat, zijn de kleuren donkerder. De tabel op de volgende pagina bevat enkele kleuren uit de lijst van beschikbare kleuren en laat zien hoe de bits gebruikt worden om de kleur te bepalen.

De lage nibble wordt gevuld met de kleur die het teken moet hebben. In de hoge nibble wordt de achtergrondkleur gezet. In bit 0 tot en met 2 wordt weer het nummer van de kleur vermeld. Afhankelijk van de stand van een byte in het geheugen wordt bit 3 gebruikt als helderheidsbit of als bit dat het teken laat knipperen. Er is een interrupt, die bestemd is om de stand van de bedoelde byte te regelen. Hiervoor wordt interrupt hex 10 gebruikt met het AH-register op hex 10.

Kleur:	Nummer:	Bits:	H:	R:	G:	B:
Zwart	0	0000				
Blauw	1	0001				*

Groen	2	0010				*			
Cyaan	3	0011				*		*	
Helderwit	15	1111		*		*		*	

Verklaring:

H = helder
R = rood
G = groen
B = blauw

In de loop der jaren is door IBM en anderen een reeks van video-adapters ontwikkeld voor toepassing in de PC. De meeste van deze adapters kunnen op verschillende resoluties ingesteld worden. De belangrijkste hiervan zijn, in chronologische volgorde:

MDA Monochroom Display Adapter tekst

CGA	Color Graphics Adapter	kleur, graphics
Hercules	Hercules Graphics Card	tekst, graphics
AT&T	AT&T/Plantronics Color-Plus	tekst, kleur, graphics
EGA	Enhanced Graphics Adapter	tekst, kleur, graphics
VGA	Video Graphics Array	tekst, kleur, graphics
MCGA	Multi-Color Graphics Array	tekst, kleur, graphics
8514/A	8514/A Display Adapter	tekst, kleur, graphics
Super VGA		tekst, kleur, graphics
XGA	eXtended Graphics Array	tekst, kleur, graphics

17.3 Grafische instellingen

De manier waarop een grafische mode in de video buffer staat, is afhankelijk van de instelling. In de grafische mode is ieder beeldpunt op het scherm adresseerbaar. Als ieder punt apart aan te sturen is, is er natuurlijk geen sprake van een tekenbyte in combinatie met een attribuutbyte, zoals in de tekst-mode. In de grafische mode wordt voor ieder beeldpunt bepaald op welke manier dat punt op het scherm moet komen.

In de Color Graphics Adapter (CGA) is onder meer een instelling mogelijk met 320 beeldpunten horizontaal en 200 beeldpunten verticaal. In totaal worden er voor een scherm dus 320 x 200 = 64000 beeldpunten gebruikt. Deze punten kunnen in vier verschillende kleuren op het scherm worden weergegeven. Iedere beschikbare kleur heeft een nummer binnen het bereik 0 tot en met 3. Om dit nummer vast te leggen, hebben we twee bits nodig. Om dit met een scherm te doen, zijn er dus 64000 x 2/8 = 16000 bytes nodig. Dat is flink wat geheugen om een grof beeld in een zeer beperkt aantal kleuren vast te leggen.

Als we deze manier van opslaan zouden gebruiken bij de VGA-adapter, waar onder meer een grafische instelling van 640 x 480 beeldpunten en 16 kleuren mogelijk is, dan zouden we hiervoor

153600 bytes nodig hebben. Hier doet zich een probleem voor. We zijn voor de videobuffer gebonden aan een maximum van 64 kilobyte. Dit is de maximale grootte van een segment. Hiervoor is een nogal ingewikkelde oplossing gevonden. De electronica van de VGA-adapter organiseert een stukje eigen geheugen dat zich op de video-adapter bevindt. Dit zijn vier arrays van 64 kilobyte die parallel met de videobuffer georganiseerd zijn. Deze vier arrays maken beurtelings gebruik van de videobuffer. De vier bits die nodig zijn om maximaal 16 kleuren te kunnen weergeven, worden over de vier arrays verdeeld. Met deze werkwijze zijn er dan geen 153600 bytes nodig maar een kwart daarvan, 38400.

De verschillende video-adapters kennen een groot aantal verschillende instellingen. Om met de grafische instellingen te kunnen werken, wordt in Turbo Pascal de unit GRAPH gebruikt. Borland levert deze unit met een aantal grafische drivers. Dit zijn stukjes programma die een specifieke video-adapter aansturen. De volgende videodrivers worden meegeleverd:

Naam bestand:	Video-adapter:
ATT.BGI	AT&T 6300 (400 lijnen)
CGA.BGI	CGA, MCGA
EGAVGA.BGI	EGA, VGA
HERC.BGI	Hercules Monochrome
IBM8514.BGI	IBM 8514/A
PC3270.BGI	IBM 3270 PC

Programma's die met de unit GRAPH werken, moeten deze bestanden kunnen vinden. Het eenvoudigst is het om ze te kopiëren naar de directory waarin je het programma maakt. Je kunt ze ook in het bestand AUTOEXEC.BAT zetten met:

append c:\tp\bgi

Hiermee kijkt DOS in de opgegeven directory als we opdracht geven naar bepaalde bestanden te zoeken.

Ook bij een gecompileerd programma, een programma met de extensie .EXE, moeten de driver-bestanden worden meegeleverd. Het is mogelijk om de drivers in het programma op te nemen. In dat geval hoeven ze niet apart meegeleverd te worden.

Niet alle grafische instellingen van de diverse video-adapters worden door de unit GRAPH ondersteund. Iedere grafische instelling die wel ondersteund wordt, heeft een naam die als constante in de unit GRAPH gedeclareerd is. Deze constanten zijn in de onderstaande tabel toegelicht.

Waar bij de kleuren de toevoeging C0, C1, C2 of C3 staat, worden verschillende palet-instellingen gebruikt. Deze paletten hebben drie verschillende kleuren plus een achtergrondkleur naar keuze. Bij EGA-Mono staat dat er 1 of 2 pagina's

beschikbaar zijn. Dit is afhankelijk van de hoeveelheid geheugen op de adapter.

Constante:	Constante Instelling:	Waarde:	Resol:	Kleur:	Pag.
CGA	CGAC0	0	320x200	C0 4	1
	CGAC1	1	320x200	C1 4	1
	CGAC2	2	320x200	C2 4	1
	CGAC3	3	320x200	C3 4	1
	CGAHi	4	640x200	2	1
MCGA	MCGA0	0	320x200	C0 4	1
	MCGAC1	1	320x200	C1 4	1
	MCGAC2	2	320x200	C2 4	1
	MCGAC3	3	320x200	C3 4	1
	MCGAMed	4	640x200	2	1
	MCGAHi	5	640x480	2	1
EGA	EGALo	0	640x200	16	4
	EGAHi	1	640x350	16	2
EGA64	EGA64Lo	0	640x200	16	1
	EGA64Hi	1	640x350	4	1
EGA-MONO	EGAMonoHi	3	640x350	2	1/2
HERC	HercMonoHi	0	720x348	2	2
ATT400	ATT400C0	0	320x200	C0 4	1
	ATT400C1	1	320x200	C1 4	1
	ATT400C2	2	320x200	C2 4	1
	ATT400C3	3	320x200	C3 4	1
	ATT400Med	4	640x200	2	1
	ATT400Hi	5	640x400	2	1
VGA	VGALo	0	640x200	16	2
	VGAMed	1	640x350	16	2
	VGAHi	2	640x480	16	1
PC3270	PC3270Hi	0	720x350	2	1
IBM8514	IBM8514Lo	0	640x480	256	1
	IBM8514Hi	0	1024x768	256	1

17.4 Lettertypen

De unit GRAPH maakt het mogelijk om met verschillende lettertypen te werken, die horizontaal of verticaal kunnen worden afgebeeld op een in te stellen grootte. Instelbare lettertypen worden ook fonts genoemd. Hiervoor worden, evenals voor de drivers, door Borland bestanden meegeleverd. Deze bestanden hebben de extensie .CHR:

BOLD.CHR
EURO.CHR
GOTH.CHR
LCOM.CHR
LITT.CHR
SANS.CHR
SCRI.CHR
SIMP.CHR
TRIP.CHR
TSCR.CHR

Ook voor de bestanden met fonts geldt, net als voor drivers, dat het programma ze moet kunnen vinden. Om het werken met de diverse fonts te vergemakkelijken, zijn er in GRAPH een aantal constanten gedefinieerd:

Naam:	Waarde:	Betekenis:
DefaultFont	0	Standaardletter met tekenveld van 8x8 bits.
TriplexFont	1	Instelbare letter.
SmallFont	2	Instelbare kleine letter.
SansSerifFont	3	Instelbare letter.
GothicFont	4	Instelbare gothische letter.
HorizDir	0	Horizontaal afdrukken.
VertDir	1	Verticaal afdrukken.
UserCharSize	0	Door programmeur te bepalen lettergrootte.

Een letter wordt gekozen door het aanroepen van de GRAPH-procedure SetTextStyle. Als parameter wordt een constante meegegeven die het gekozen lettertype representeert, en een constante voor de richting en de afmeting van de letter.

Als de programmeur de afmeting van de letters bepaalt, dan wordt voor de afmeting de constante UserCharSize meegegeven. In dat geval moet wel met SetUserSize de afmeting van de letter ingesteld zijn. SetUserSize krijgt een viertal parameters van het type Word mee. De eerste twee worden gebruikt om de breedte in te stellen. De eerste vermenigvuldigt en de tweede deelt, zodat er een breuk ontstaat. Het tweede stel wordt op dezelfde manier gebruikt, maar nu voor de hoogte. SetUserSize(1,2,1,1) stelt een letter in op de halve breedte, terwijl de hoogte ongewijzigd blijft.

Je kunt de tekstinstellingen ook weer teruglezen. Hiervoor wordt de GRAPH-procedure GetTextSettings gebruikt. Deze procedure krijg als VAR-parameter een variabele van het type TextSettingsType mee. Dit type is in de unit GRAPH gedefinieerd:

```
TextSettingsType = Record
   Font     : Word;
   Direction: Word;
   CharSize : Word;
   Horiz    : Word;
   Vert     : Word;
END;
```

Bij terugkomst uit de procedure GetTextSettings staan in het meegezonden record de gegevens over de geldende tekstinstellingen. Ook het aantal beeldpunten dat een bepaalde tekst in beslag neemt, kan opgevraagd worden. Hiervoor wordt GetTextSize gebruikt. GetTextSize krijgt als parameter een string mee met daarin de tekst, die onderzocht moet worden.

17.5 Werken met kleuren

De manier waarop kleuren georganiseerd zijn in CGA en MCGA is rechtlijnig en gemakkelijk te begrijpen. Afhankelijk van de gekozen resolutie zijn er één of twee bits beschikbaar om de kleur van een beeldpunt op het scherm vast te leggen. De getalswaarde van die bits wijst naar een gekozen palet. In de hoge resolutie beschikken deze adapters over een zwarte achtergrond en een voorgrondkleur. In de lagere resoluties kan er gekozen worden tussen vier paletten:

```
|Palet:          |Beschikbare kleuren:                                  |
|C0              |zwart, lichtrood, licht groen, geel                   |
|C1              |zwart, licht cyaan, licht magenta, wit                |
|C2              |zwart, rood, groen, bruin                             |
|C3              |zwart, cyaan, magenta, lichtgrijs                     |
```

Bij EGA en VGA ligt dit anders. De unit GRAPH ondersteunt hier zestien kleuren. Deze kleuren staan in een record van het voorgedefinieerde type PaletteType. De vorm van dit record is:

PaletteType = Record

```
   Size  : Byte;
   Colors: Array [0.. MaxColors] of ShortInt;
     END;
```

In de verschillende elementen van de array Colors worden de kleurnummers gezet. Bij de start van een programma ziet het palet er als volgt uit:

Element:	Kleur:
0	0
1	1
2	2
3	3
4	4
5	5
6	20
7	7
8	56
9	57
10	58
11	59
12	60
13	61
14	62
15	63

De index van de array komt overeen met het lijstje kleuren uit hoofdstuk 2. De waarde van het element Colors[4] wijst naar rood.

Als je een instelling van een bepaald grafisch scherm kiest, dan zal dit scherm zich altijd manifesteren in kleur 0. Je zou dit kunnen zien als een achtergrondkleur. Als je het palet niet zou kunnen veranderen, zou het scherm dus altijd zwart zijn. De procedure SetBkGround(kleur) levert hiervoor de oplossing. SetBkGroundColor(9) zet het kleurnummer uit element 9 in element 0, waardoor het scherm lichtblauw wordt. Veranderingen in het palet hebben gevolgen voor de kleuren op het scherm.

Nu hebben we wel een lichtblauwe achtergrond, maar als we daar met zwart op willen tekenen, dan beschikken we niet meer over zwart omdat kleurnummer 0 zich niet meer in het palet bevindt. Maar geen nood. Ook hier is een oplossing voor, want SetPalette(9,0) zet een 0 in het negende element.

Als we nu SetColor(9) aanroepen, selecteren we element 9 als kleur om te tekenen. Element 9 wijst nu naar de kleur zwart in plaats van naar lichtblauw. We moeten nu dus element 9 kiezen om met zwart te kunnen tekenen, in plaats van element 0.

In de grafische mode levert de unit GRAPH ons, mits er een EGA- of VGA-adapter geïnstalleerd is, een palet met zestien kleuren die geselecteerd zijn uit de vierenzestig kleuren die EGA levert. Je kunt het aantal ingangen van de geïnstalleerde video-adapter te weten komen door de GRAPH-functie GetPaletteSize aan te roepen.

Met VGA is het nog wat ingewikkelder. Deze adapter bevat een zogenaamde Video DAC (Video Digital Analog Converter). Deze converter heeft 64 ingangen die waarden bevatten voor instructies over de mengverhouding van de te presenteren kleuren. Per ingang zijn er waarden voor rood, groen en blauw opgeslagen. De waarde die aan elk van de kleuren gegeven kan worden, varieert van 0 tot en met 63.

Door de onderlinge waarden van deze kleuren te wijzigen, wordt de mengverhouding van de kleuren gewijzigd. De reden dat we hier een omgekeerde ontwikkeling zien, ligt in de grotere mogelijkheden die een analoge monitor heeft bij het weergeven van kleur. In de systemen die gebruik maken van een analoge monitor kunnen we de hoeveelheid blauw, groen en rood in een kleur regelen. Met een digitale monitor kan rood, groen of blauw alleen maar aan- of uitgezet worden. Op analoge monitoren zijn dus rijkere kleurschakeringen mogelijk.

De unit GRAPH heeft een procedure SetRGBPalette. Met de aanroep:

SetRGBPalette(20,63,63,63);

zal kleurnummer 20, dat in element 6 van het palet staat, helderwit opleveren in plaats van bruin. Op deze manier kan er bij VGA een palet van 16 kleuren worden samengesteld, dat gekozen wordt uit 64 beschikbare kleuren. Elk van deze kleuren kan ook weer afzonderlijk worden samengesteld. Dit betekent dat we kunnen kiezen uit ruim 262.000 kleurnuanceringen.

In de unit GRAPH zijn voor de kleuren een aantal constanten gedefinieerd. Deze constanten werken natuurlijk alleen goed als er geen veranderingen in het palet zijn aangebracht.

In de tabellen op de volgende pagina's staan de namen voor de zestien paletingangen. Zoals we gezien hebben, bevatten deze ingangen een waarde die wijst naar het EGA- kleurenregister:

Donkere kleuren:

Constante:	Waarde:	Kleur:
Black	0	Zwart
Blue	1	Donkerblauw
Green	2	Donkergroen
Cyan	3	Cyaan
Red	4	Rood
Magenta	5	Magenta
Brown	6	Bruin
LightGray	7	Lichtgrijs

Lichte kleuren (in tekst-mode alleen te gebruiken als voorgrondkleur):

Constante:	Waarde:	Kleur:
DarkGray	8	Donkergrijs =
		lichtzwart
LightBlue	9	Lichtblauw
LightGreen	10	Lichtgroen
LightCyan	11	Lichtcyaan
LightRed	12	Lichtrood
LightMagenta	13	Lichtmagenta
Yellow	14	Geel
White	15	Wit

* Om een leesteken in de tekst-mode te laten knipperen moet er 128 bij de kleurwaarde opgeteld worden.

Voor de standaardwaarden van het EGA-palet zijn ook constanten gedefinieerd:

Naam:	Waarde:
EGABlack	0
EGABlue	1
EGAGreen	2
EGACyan	3
EGARed	4
EGAMagenta	5
EGABrown	20
EGALightGray	7
EGADarkGray	56
EGALightBlue	57
EGALightGreen	58

EGALightCyan	59	
EGALightRed	60	
EGALightMagenta	61	
EGAYellow	62	
EGAWhite	63	

17.6 Constanten voor vulpatronen

In de unit GRAPH kun je vlakken vullen met een voorgedefinieerd patroon of met een patroon dat je zelf gemaakt hebt. Deze patronen worden ingesteld met de GRAPH-procedure SetFillStyle. SetFillStyle krijgt als parameters het nummer van een patroon dat gebruikt moet worden, en het nummer van de kleur waarin het patroon moet worden afgedrukt mee.

De volgende tabel geeft een overzicht van de patronen:

Naam:	Waarde:	Resultaat:
EmptyFill	0	Gebruikt kleur 0 uit het palet.
SolidFill	1	Maakt een leeg vlak zonder patroon in de aangegeven kleur.
LineFill	2	Maakt een patroon van streepjes.
LtSlashFill	3	Maakt een patroon van dunne diagonale lijnen van rechtsboven naar linksonder.
SlashFill	4	Maakt een patroon van dikke diagonale lijnen van rechtsboven naar linksonder.
BkSlashFill	5	Maakt een patroon van dunne diagonale lijnen van linksboven naar rechtsonder.
LtBkSlash	6	Maakt een patroon van dikke diagonale lijnen van linksboven naar rechtsonder.
HatchFill	7	Maakt een ruitjespatroon in dunne lijnen.
XHatchFill	8	Maakt een diagonaal ruitjespatroon in dikke lijnen.
InterleaveFill	9	Maakt een rasterpatroon.
WideDotFill	10	Maakt een patroon met breed gespreide punten.
CloseDotFill	11	Maakt een patroon van dicht bij elkaar liggende punten.
UserFill	12	Is een door de programmeur gemaakt patroon.

Je kunt zelf een patroon maken door een variabele te declareren van het type FillPatternType. Dit in GRAPH gedefinieerde type is een array van acht bytes. Hiermee kan in een vlakje van acht bij acht beeldpunten een patroon gemaakt worden. Als een bit op 0 staat, wordt het beeldpunt afgebeeld in de achtergrondkleur. Staat de bit op 1, dan wordt het beeldpunt afgebeeld in de kleur die ingesteld is voor het afdrukken van het vulpatroon. Er kan ook gekeken worden welk patroon ingesteld is.
De GRAPH-procedure GetFillSettings krijgt als VAR-parameter een variable van het type FillSettingsType mee. Dit in GRAPH gedefinieerde record heeft de volgende vorm:

FillSettingsType = Record

```
        Pattern: Word;
   Color  : Word;
     END;
```

Bij terugkomst uit GetFillSetting staat in het veld Pattern het nummer van het ingestelde patroon en in het veld Color de hiervoor ingestelde kleur.

Ook het patroon zelf kan opgevraagd worden met GetFillPattern. GetFillPattern krijgt als VAR-parameter een variabele van het type FillPatternType mee. Dit type is in GRAPH gedefinieerd als Array [1..8] of Byte.

De variabele van het type FillPatternType wordt meegegeven aan de GRAPH-procedure SetFillPattern. Deze procedure zorgt ervoor dat de vulstijl gekoppeld is aan de waarde uit de tabel van UserFill. Tot er een ander patroon gekoppeld wordt aan UserFill, kan het door de programmeur vervaardigde patroon nu ingesteld worden met SetFillStyle.

17.7 Constanten voor lijnpatronen

Ook voor het tekenen van lijnen zijn in GRAPH een aantal constanten gedefinieerd. Lijnen kunnen ingesteld worden op een verschillend patroon en dikte:

Naam:	Waarde:	Betekenis:
SolidLn	0	Ononderbroken lijn.
DottedLn	1	Lijn van puntjes.
CenterLn	2	Lijn van afwisselend kort en lang streepje.
DashedLn	3	Lijn van blokjes.
UserBitLn	4	Lijn gemaakt door de programmeur.

De lijnen worden als volgt ingesteld met de GRAPH-procedure:

SetLineStyle(LijnStijl:Word;Patroon:Word;Dikte:Word)

Voor LijnStijl wordt dan één van de constanten meegegeven. Als aan SetLineStyle de waarde UserBitLn wordt meegegeven, dan moet ook de parameter Patroon een waarde krijgen. In de andere gevallen kan daaraan de waarde 0 worden gegegeven. Bij UserBitLn wordt een Word, ofwel 16 bits, meegegeven. Als een bit op 0 staat, wordt de achtergrondkleur in het beeldpunt gezet. Indien de bit de waarde 1 heeft, wordt het beeldpunt geplaatst in de kleur die ingesteld is met SetColor.

Het is ook mogelijk om de ingestelde lijnstijl weer op te vragen. Hiervoor hebben we de GRAPH-procedure GetLineSettings. Deze procedure krijgt als VAR-parameter een variabele van het type LineSettingsType mee. Dit type is in GRAPH gedefinieerd:

```
LineSettingsType = Record
   LineStyle: Word;
   Pattern  : Word;
   Thickness: Word;
END;
```

Bij terugkeer uit GetLineSettings staan de gezochte gegevens in

het meegezonden record. Voor de dikte van de lijn is een tweetal constanten gedefinieerd:

Naam:	Waarde:	Betekenis:
NormWidth	0	Gewone dikte
ThickWidth	1	Dubbele dikte

17.8 De instellingen programmeren

Het volgende programma dat we in Turbo Pascal gaan maken laat zien hoe bepaald wordt met welke driver gewerkt moet worden. Het programma moet de mogelijke grafische instellingen van de in de machine geïnstalleerde video-adapter afgaan en op het scherm melden welke instelling gebruikt wordt. Het programma moet bovendien, indien mogelijk, heen en weer schakelen tussen verschillende pagina's.

Ook in dit programma zijn weer een functie en een procedure gemaakt die we waarschijnlijk nog wel eens nodig kunnen hebben. Vandaar dat we ze opnemen in de unit DIVERSEN. De procedure Schrijf verdeelt het scherm in een ingesteld aantal horizontale en verticale tekens. Onafhankelijk van de gebruikte resolutie zet Schrijf een tekst op de bedoelde plaats op het scherm.

De functie Paginas retourneert het aantal schermpagina's van de gekozen instelling op de video-adapter. Let wel op: omdat de procedure Schrijf de procedures GetMaxX en GetMaxY van de unit GRAPH aanroept, moet in de USES-regel van de unit DIVERSEN de unit GRAPH opgenomen worden. Dit gaat als volgt:

```
CONST
   AANTAL_REGELS : Word = 20;
   AANTAL_LETTERS: Word = 80;

PROCEDURE Schrijf(X,Regel:Word; S:String);
VAR
   Y: Word;

BEGIN
   X := (GetMaxX DIV AANTAL_LETTERS) * X;
   Y :=  GetMaxY DIV AANTAL_REGELS * Regel;
   OutTextXY(X, Y, S)
END;

FUNCTION Paginas(Driver:Integer):Byte;
VAR
   MODE: Byte;

BEGIN
   MODE := GetGraphmode;
   Paginas := 1;
   CASE Driver OF
      EGA: CASE MODE OF
         0: Paginas := 4;
         1: Paginas := 2;
      END;
      HERCMONO: IF MODE = 0 THEN Paginas := 2;
      VGA: IF MODE IN [0,1] THEN Paginas := 2
   END
END;
```

Regels:Toelichting:

[1]1-3Declareer een getypeerde constante om het aantal regels op een grafisch scherm aan te geven.
[2]4-11Procedure Schrijf(X,Regel:Word;S:String).
5-6Declareer een lokale variabele Y.
[3] 8-9Bereken een positie op het scherm.
[4]10 Zet Regel op de berekende positie.
[5]12-26Function Paginas(Driver:Integer):Byte.
13-14Declareer een lokale variabele.
[6]16 Vraag de instelling op.
[7]17 Geef Paginas de waarde 1.
18-25Geef alleen bij bepaalde drivers in een bepaalde instelling Paginas een andere waarde.

Toelichting:

[1]Als je een tekst of afbeelding op een bepaalde positie op een grafisch scherm wilt plaatsen, kun je niet met absolute waarden voor de schermcoördinaten werken. Het aantal mogelijke X- en Y-coördinaten van een VGA-scherm dat ingesteld is op 640 x 480 beeldpunten, is natuurlijk veel groter dan bij een CGA-scherm dat ingesteld is op een formaat 320 x 200. Als in Y een waarde 100 staat, dan zou deze coördinaat wijzen naar een positie in het bovenste kwart van het scherm. Bij CGA zou Y wijzen naar het midden van het scherm. In dit voorbeeld geven we aan AANTAL_REGELS het aantal tekstregels en aan AANTAL_LETTERS het aantal letters op een regel.

[2]Deze gegevens worden gebruikt door de procedure Schrijf.

[3]Voor de berekening van de X-coördinaat wordt de functie GetMaxX uit GRAPH aangeroepen. Deze functie retourneert het aantal horizontale beeldpunten van het ingestelde grafische scherm. Dit aantal wordt gedeeld door de waarde in AANTAL_LETTERS. Op die manier krijg je het aantal beeldpunten per letter. De uitkomst hiervan wordt vermenigvuldigd met de waarde in de parameter X. De uitkomst hiervan wijst naar een relatief X-coördinaat. De Y-coördinaat wordt op dezelfde wijze berekend. Alleen wordt hier niet GetMaxX aangeroepen, maar GetMaxY.

[4]Met de GRAPH-procedure OutTextXY wordt de inhoud van de parameter S op het scherm gezet. In de grafische mode maken we geen gebruik van Write of Writeln. In de grafische mode kunnen we gebruik maken van OutTextXY of van OutText. Bij OutTextXY geven we behalve de af te beelden tekst ook de X- en Y-coördinaten mee. Bij OutText verschijnt de tekst op de plaats waar de onzichtbare cursor zich bevindt.

[5]De functie Paginas retourneert het aantal mogelijke schermpagina’s bij een bepaalde instelling van de video-adapter. Deze functie maakt gebruik van de gegevens in de handleiding van Borland. Het is natuurlijk eleganter om het aantal mogelijke pagina’s in het geheugen op te zoeken. Het ROM BIOS levert wel

een dergelijke service-routine. Deze is echter alleen toegankelijk voor de PS/2 en voor machines die uitgerust zijn met een VGA-adapter. Vandaar dat we de gegevens van Borland gebruiken. Voor degenen die de functie willen ombouwen, zodat het geheugen direct uitgelezen wordt, volgen hieronder de benodigde gegevens:

De serviceroutine wordt aangeroepen door interrupt hex 10. Bij het aanroepen moet het register AH op hex 1B gezet worden. Bij terugkeer staat in DI het segment en in ES de offset van de videogegevens. Tel nu bij de offset hex 29 op. Zo krijgen we het adres van de byte waarin het aantal pagina's staat dat bij het ingestelde grafische scherm mogelijk is.

[6]De instelling van de video-adapter kan opgevraagd worden met de functie GetGraphMode uit de unit GRAPH.

[7]Uit de functie Paginas blijkt dat alleen bepaalde modes bij EGA, Hercules en VGA beschikken over meer schermpagina's. Vandaar dat Pagina's gewoonlijk de waarde 1 retourneert. Alleen als er een video-adapter geïnstalleerd is, die bovendien nog in een bepaalde mode staat, wordt een andere waarde geretourneerd.

Het programma GRAPH_1 luidt als volgt:

```
{$X+}
PROGRAM GRAPH_1;
USES CRT, GRAPH, DIVERSEN;

VAR
   GRAFISCHE_DRIVER, INSTELLING: Integer;
   LAAGSTE_INSTELLING, HOOGSTE_INSTELLING: Integer;
   ZIN: String;
   I: Byte;


PROCEDURE WisselPagina;
BEGIN
   Schrijf
   (30,14,'Druk op toets voor andere pagina...');
   SetActivePage(1);
   SetColor(Red);
   Schrijf(30,10,'Dit is pagina 2');
   Schrijf(30,14,'Druk op toets voor pagina 1...');
   ReadKey;
   SetVisualPage(1);
   SetActivePage(0);
   SetColor(White);
   Schrijf
   (30,14,'Druk op toets voor andere pagina...');
   SetColor(Blue);
   Schrijf
   (30,14,'Terug op de 1e pagina. Druk op een toets');
   ReadKey;
   SetVisualPage(0);
   ReadKey
END;

BEGIN
   GRAFISCHE_DRIVER := Detect;
   InitGraph(GRAFISCHE_DRIVER,INSTELLING,'');
   IF GraphResult <> GrOK THEN Halt;
   GetModeRange(GRAFISCHE_DRIVER,LAAGSTE_INSTELLING,
                HOOGSTE_INSTELLING);
   FOR I := LAAGSTE_INSTELLING TO HOOGSTE_INSTELLING DO
   BEGIN
      SetGraphMode(I);
      SetBkColor(White);
      SetColor(Blue);
      Schrijf(30,10,GetModeName(I));
      Str(I,ZIN);
```

```
      SetTextStyle(SmallFont,HorizDir,4);
      Schrijf(30,11,'INSTELLING      : '+ZIN);
      Str(Paginas(GRAFISCHE_DRIVER),ZIN);
      Schrijf(30,12,'Aantal pagina's : '+ZIN);
      IF Paginas(GRAFISCHE_DRIVER) > 1 THEN WisselPagina
      ELSE
      ReadKey
   END;
   CloseGraph
END.
```

Regels:Toelichting

3 Gebruik behalve CRT en GRAPH ook de unit DIVERSEN.

[0]4-8 Declareer de globale variabelen van het programma.

9-29Procedure WisselPagina.

[14]11-12 Zet een mededeling op de eerste pagina.

[15]13 Maak de volgende pagina actief.

[16]14-16 Schrijf tekst naar de tweede pagina.

[17]17 Wacht tot er bij de eerste pagina een toets wordt ingedrukt.

[18]18 Maak de tweede pagina zichtbaar.

[19]19-22Maak van de eerste pagina de actieve pagina en verwijder de onderste mededeling uit de eerste pagina.

[20]23-25 Zet een nieuwe mededeling op de eerste pagina.

[21]26 Wacht op het indrukken van een toets om terug te gaan naar de eerste pagina.

[22]27 Maak de eerste pagina weer zichtbaar.

[23]28 Wacht op het indrukken van een toets. Keer daarna terug naar het hoofdprogramma.

30-52Hoofdprogramma.

[1]31 Geef DRIVER de waarde Detect.

[2]32 Roep InitGraph aan om het juiste driverbestand met het programma te verbinden.

[3]33 Stop de uitvoering van het programma als er iets fout gegaan is.

[4]34-35Roep GetModeRange aan om het aantal mogelijke instellingen van de video-adapter te weten te komen.

[5]36-50Ga de FOR-lus in die de verschillende mogelijke instellingen van de video-adapter doorloopt.

[6]38 Roep SetGraphMode aan om de adapter in te stellen.

[7]39 Roep SetBkColor aan om het scherm een kleur te geven.

[8]40 Kies blauw als kleur voor de uitvoer naar het scherm.

[9] 41 Zet het resultaat van de aanroep van GetModeName op het scherm.

42 Zet de geldige instelling in ZIN.

[10]43 Roep SetTextStyle aan en kies SmallFont als letter die horizontaal op het scherm moet worden gezet.

[11]44Schrijf ZIN naar het scherm.

[12]45-46Roep de functie Paginas uit de unit DIVERSEN aan en zet de geconverteerde uitkomst in ZIN. Schrijf ZIN naar het scherm.

```
   [13]47-49Roep Paginas nog een keer aan. Als de uitkomst groter
          dan 1 is, ga dan naar Wisselpagina om een andere pagina
          te laten zien. Als het aantal pagina's 1 is, wacht  dan
          op het indrukken van een toets.
   [24]51 Roep CloseGraph aan om alles weer in de oorspronkelijke
          staat terug te brengen.
```

Toelichting:

```
   [0]De gedeclareerde globale variabelen zijn bestemd om de
volgende gegevens te bevatten:
```

GRAFISCHE_DRIVER: nummer van de geïnstalleerde adapter.
INSTELLING: waarde van de gekozen instelling.

[1]Als je een grafisch programma algemeen toepasbaar wilt maken, laat dan de unit GRAPH uitzoeken welke video-adapter geïnstalleerd is. Als je van tevoren weet dat het programma alleen maar op een machine met een CGA-adapter hoeft te draaien, kun je natuurlijk direct voor CGA kiezen. Een keuze voor de voorgedefineerde variabele Detect in het begin van je grafische programma ligt echter meer voor de hand. Als DRIVER de waarde Detect krijgt en deze waarde doorgegeven wordt aan InitGraph, dan selecteert InitGraph de driver die bij de gebruikte configuratie past.

[2]InitGraph wordt aangeroepen om het juiste driverbestand met het programma te verbinden. InitGraph kiest, als de waarde van Detect wordt doorgegeven, altijd de hoogst mogelijke resolutie.

```
Bij terugkeer uit InitGraph staat in de parameter  INSTELLING  de
waarde van de ingestelde mode. In de derde  pagina  kan  een  pad
gezet worden naar  de  directory  waar  zich  de  driverbestanden
bevinden. Pas wel op: bij iedere aanroep van InitGraph wordt door
de procedure een gegevensset op  de  heap  gezet.  Als  InitGraph
gedurende de  uitvoering  van  het  programma  meer  keren  wordt
aangeroepen, zal er telkens een gegevensset bij geplaatst  worden
en zal de heap snel vollopen.
```

```
   [3]Als er iets fout is gegaan, staat de foutcode in de
variabele GraphResult. De betekenis van de code in GraphResult staat in de onderstaande tabel:
```

Waarde:	Constante:	Omschrijving:
0	GrOK	Geen fout gevonden.
-1	GrNoInitGraph	BGI driverbestand niet genstalleerd.
-2	GrNotDetected	Geen grafische adapter te vinden.
-3	GrFileNotFound	Bestand van driver niet gevonden.
-4	GrInvalidDriver	Bestand van driver is beschadigd.
-5	GrNoLoadMem	Niet genoeg geheugen om driver in te laden.
-6	GrNoScanMem	Niet genoeg geheugen om opgeslagen gegevens te

		verwerken.
-7	GrNoFloodMem	Niet genoeg geheugen voor vervulling opdracht.
-8	GrFontNotFound	Bestand met font niet gevonden.
-9	GrNoFontMem	Niet genoeg geheugen om lettertype te laden.
-10	GrInvalidMode	Gekozen instelling past niet bij driver.
-11	GrError	Niet-specifieke foutmelding.
-12	GrIOError	In- of uitvoerfout van graphicsbestand.
-13	GrInvalidFont	Beschadigd fontbestand.
-14	GrInvalidFontNum	Onjuist lettertype-nummer.

Je kunt ook nog de bij de fout behorende tekst af laten drukken door de GRAPH-functie GetErrorMessage aan te roepen. Deze functie krijgt als parameter de foutcode door en retourneert de Engelse toelichting. Voor Nederlandse programma's kun je het beste, als je een dergelijke functie wilt gebruiken, een soortgelijke Nederlandstalige functie schrijven en die aanroepen in plaats van GraphErrorMessage.

[4]Het aantal instellingen dat gekozen kan worden, verschilt sterk bij de verschillende video-adapters. De GRAPH-procedure GetModeRange retourneert het bereik van de geldige instellingen van de geïnstalleerde video-adapter. GetModeRange krijgt als eerste parameter de variabele GRAFISCHE_DRIVER mee en vervolgens de variabelen LAAGSTE_INSTELLING en HOOGSTE_INSTELLING. GetModeRange handelt deze laatste parameters af als VAR-parameters. Bij terugkomst staat in LAAGSTE_INSTELLING de waarde van de laagste mode van de geïnstalleerde video-adapter en in HOOGSTE_INSTELLING de hoogste. Als je alleen de hoogst mogelijke instelling nodig hebt, kun je de GRAPH-functie GetMaxMode aanroepen. Deze functie retourneert het maximale aantal mogelijke instellingen van de geïnstalleerde video-adapter.

[5]De FOR-lus die nu ingegaan wordt, zet I op LAAGSTE_INSTELLING. Dan wordt I net zolang verhoogd tot de waarde van HOOGSTE_INSTELLING bereikt is.

[6]SetGraphMode stelt de video-adapter in. SetGraphMode krijgt als parameter I mee. Hierin staat immers de gewenste instelling.

[7]SetBkColor(White) zorgt er voordat de beeldpunten wit op het scherm gezet worden. SetBkColor verandert het palet waarmee gewerkt wordt. Na deze aanroep wijst de eerste ingang van het palet naar wit. Als je met een VGA- of EGA-adapter werkt, dan heb je 16 kleuren ter beschikking. De instelwaarde voor de eerste ingang is dan zwart. Met SetBkColor wijst de eerste ingang nu naar wit. Je kunt de ingestelde kleur ook weer opvragen. De GRAPH-functie GetBkColor retourneert het kleurnummer dat in element 0 van het palet staat.

[8]Vervolgens wordt met SetColor(Blue) voor uitvoer naar het scherm de kleur blauw gekozen. Hier had ook SetColor(1) kunnen staan. Blauw heeft kleurnummer 1.

[9]De procedure GetModeName uit de unit GRAPH krijgt als parameter I mee. In I staat de waarde van de gekozen instelling. Aan de hand van de geïnstalleerde video-adapter en de doorgegeven instelling, retourneert GetModeName de naam van de driver en de afmeting van de gekozen instelling. Met de aanroep van de procedure Schrijf wordt de uitkomst van GetModeName op het scherm gezet.

[10]SetTextStyle verbindt vervolgens het programma met het gekozen fontbestand. In dit geval wordt de GRAPH-constante HorizDir meegegeven ten teken dat we de tekst horizontaal op het scherm willen tonen. Zouden we de constante VertDir meegegeven hebben, dan zouden de teksten verticaal op het scherm verschijnen.

[11]De inhoud van de variabele ZIN wordt vervolgens in het zojuist gekozen lettertype door de procedure Schrijf op het scherm geplaatst.

[12]Het aantal mogelijke pagina's van de gekozen instelling wordt geretourneerd door de functie Paginas. De uitkomst wordt geconverteerd naar ZIN. De procedure Schrijf zet de gevonden waarde op het scherm.

[13]Opnieuw wordt Paginas aangeroepen. Als de uitkomst groter dan 1 is, springen we naar de procedure WisselPagina. Is de uitkomst 1 dan wachten we op het indrukken van een toets. Daarna keren we terug naar het begin van de FOR-lus.

[14]In de procedure WisselPagina plaatsen we op het scherm de mededeling dat er op een toets gedrukt moet worden voor de volgende pagina. Als we in de procedure WisselPagina zijn, dan is het duidelijk dat er een volgende pagina aanwezig is.

[15]De volgende stap is het aanroepen van SetActivePage. SetActivePage schakelt over naar een geselecteerde pagina zonder deze zichtbaar te maken. De uitvoer komt echter wel terecht in de door SetActivePage geselecteerde pagina. Zo'n pagina wordt geselecteerd door het nummer als parameter mee te geven. Het eerste nummer is 0, het tweede is 1, enzovoort.

[16]De voorgrondkleur van de onzichtbare pagina wordt op rood gezet. Schrijf zet de mededeling "Dit is pagina 2" op het scherm en vraagt om een toetsindruk.

[17]De ReadKey die nu volgt, wacht op een toetsindruk bij het nog zichtbare scherm. Dit is het eerste scherm dat we vulden.

[18]Pas nadat er op een toets gedrukt is, maken we de tot nu toe verborgen pagina zichtbaar door het aanroepen van SetVisualPage(nummer). Dit wisselen van pagina's gaat razendsnel. Een voordeel van deze techniek is dat je het opbouwen van de schermen niet kunt zien. Met name bij het maken van animaties is dit erg nuttig.

[19]We maken vervolgens de eerste pagina weer actief en schrijven de mededeling dat op een toets gedrukt moet worden voor een ander pagina nog een keer naar het scherm. De mededeking krijgt dezelfde kleur als de achtergrondkleur zodat deze op het scherm niet zichtbaar is.

[20]We kiezen nu voor blauw en schrijven op het scherm dat we terug zijn op de eerste pagina en dat er op een toets gedrukt moet worden.

[21]ReadKey wacht op een toetsindruk bij de tweede pagina.

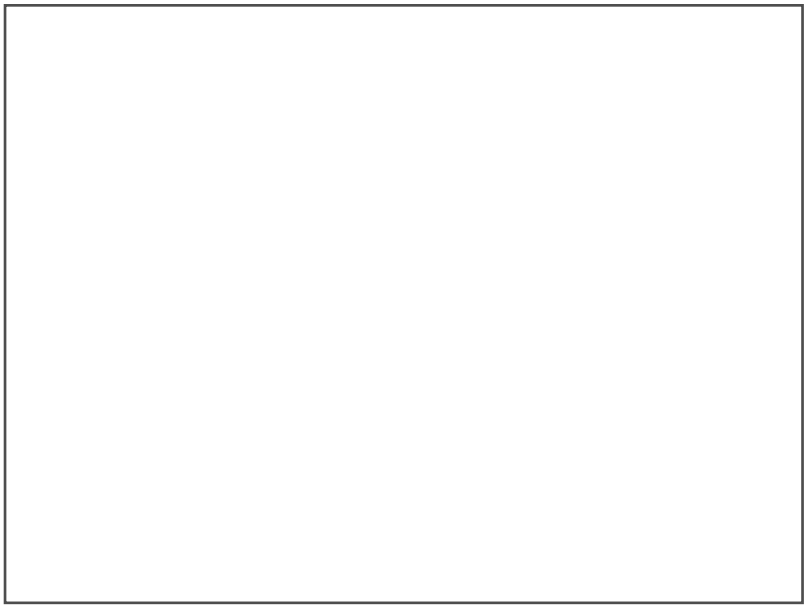
[22]Daarna wordt de oorspronkelijke pagina weer zichtbaar gemaakt.

[23]Na een toetsaanslag keren we terug in het hoofdprogramma.

[24]Als de lus doorlopen is, moet CloseGraph aangeroepen worden om de verbinding tussen het programma en de driver te verbreken en de situatie weer te herstellen in de toestand als voor de aanroep van InitGraph.

17.9 Grafische toepassingen

Het nu volgende programma GRAPH2.PAS opent een achttal grafische venstertjes op het scherm en demonstreert in ieder van die venstertjes één of meer mogelijkheden van de unit GRAPH.



Afbeelding 16

```
PROGRAM GRAPH_2;
USES CRT, GRAPH, DIVERSEN;
CONST
   SCRIPT_FONT  = 5;
   SIMPLEX_FONT = 6;

FUNCTION BinWord(BitPatroon:String):Word;
VAR
   I   : Byte;
   B, J: Word;

BEGIN
   B := 0;
   J := 1;
   IF Length(BitPatroon) >= 16 THEN
   BitPatroon := Copy(BitPatroon,1,16)
   ELSE
   FOR I := Length(BitPatroon) TO 16 DO
   BitPatroon := '0' + BitPatroon;
   FOR I := Length(BitPatroon) DOWNTO 1 DO
   BEGIN
      IF NOT (BitPatroon[I] IN ['0','1']) THEN
      BEGIN
        BinWord := 0;
        Exit
      END
      ELSE
      BEGIN
        B := B +(Ord(BitPatroon[I])-48) * J;
        J := J * 2 ;
      END
   END;
   BinWord := B
END;

PROCEDURE Lijnen;
CONST
   DELER: Byte = 6;
VAR
   VIEWREC: ViewPortType;

BEGIN
   SetFillStyle(SolidFill,Blue);
   Bar(0,0,GetMaxX,GetMaxY);
   SetColor(White);
   GetViewSettings(VIEWREC);
```

```
   WITH VIEWREC DO
   BEGIN
      Line((X2-X1) DIV DELER,0,(X2-X1) DIV DELER,Y2);
      SetLineStyle(DottedLn,0,NormWidth);
      Line((X2-X1) DIV DELER*2,0,
          (X2-X1) DIV DELER*2,Y2);
      SetLineStyle(CenterLn,0,NormWidth);
      Line((X2-X1) DIV DELER*3,0,
          (X2-X1) DIV DELER*3,Y2);
      SetLineStyle(DashedLn,0,ThickWidth);
      Line((X2-X1) DIV DELER*4,0,
          (X2-X1) DIV DELER*4,Y2);
      SetLineStyle(UserBitLn,BinWord('000001110000'),
                   ThickWidth);
      Line((X2-X1) DIV DELER*5,0,
          (X2-X1) DIV DELER*5,Y2);
      SetLineStyle(SolidLn,0,NormWidth)
   END
END;

PROCEDURE Rechthoeken;
VAR
   VIEWREC  : ViewPortType;
   HOR, VERT: Word;
   PATROON  : FillPatternType;

BEGIN
   SetFillStyle(SolidFill,Yellow);
   Bar(0,0,GetMaxX,GetMaxY);
   GetViewSettings(VIEWREC);
   SetColor(Blue);
   SetLineStyle(SolidLn,0,NormWidth);
   WITH VIEWREC DO
   BEGIN
      HOR := X2 - X1;
      VERT := Y2 - Y1;
      PATROON[1] := BinWord('00000000');
      PATROON[2] := Binword('00000000');
      PATROON[3] := BinWord('00111100');
      PATROON[4] := BinWord('00100100');
      PATROON[5] := BinWord('00100100');
      PATROON[6] := BinWord('00111100');
      PATROON[7] := BinWord('00000000');
      PATROON[8] := BinWord('00000000');
      SetFillPattern(PATROON,LightGreen);
```

```
      RecTangle(HOR * 10 DIV 100,VERT * 10 DIV 100,
                HOR - (HOR * 10 DIV 100),
                VERT - (VERT * 10 DIV 100));
      FloodFill(HOR DIV 2,VERT DIV 2,GetColor)
   END
END;

PROCEDURE Bars;
VAR
   VIEWREC  : ViewPortType;
   HOR, VERT: Word;

BEGIN
   SetFillStyle(SolidFill,LightRed);
   Bar(0,0,GetMaxX,GetMaxY);
   GetViewSettings(VIEWREC);
   WITH VIEWREC DO
   BEGIN
      HOR  := X2 - X1;
      VERT := Y2 - Y1;
      SetFillStyle(LtSlashFill,lightBlue);
      Bar(10 * HOR DIV 100,20 * VERT DIV 100,
          HOR - (HOR * 10 div 100),40 * VERT DIV 100);
      SetFillStyle(SlashFill,White);
      Bar(25 * HOR DIV 100,60 * VERT DIV 100,
          HOR - (HOR * 25 div 100),90 * VERT DIV 100)
   END
END;

PROCEDURE Bar_3D;
CONST
   AANTAL = 5;
VAR
   VIEWREC: ViewPortType;
   HOR, VERT, BREED, START, BODEM, DIEP: Word;
   I: Byte;

BEGIN
   SetFillStyle(SolidFill,Blue);
   Bar(0,0,GetMaxX,GetMaxY);
   GetViewSettings(VIEWREC);
   Randomize;
   WITH VIEWREC DO
   BEGIN
      HOR  := X2 - X1;
      VERT := Y2 - Y1;
```

```
      BREED := HOR DIV (AANTAL+1);
      BODEM := VERT - (VERT * 15 DIV 100);
      DIEP  := BREED * 20 DIV 100;
      START := 10 * HOR DIV 100;
      FOR I := 1 TO AANTAL DO
      BEGIN
         REPEAT
            SetColor(Random(GetMaxColor))
         UNTIL GetColor <> Blue;
         SetFillStyle(Random(11),Random(GetMaxColor));
         Bar3D(START,Random(VERT),START+BREED,BODEM,
               DIEP,TopOn);
         Inc(START,BREED)
      END
   END
END;

PROCEDURE Cirkel;
CONST
   AFSTAND: Byte = 6;
VAR
   VIEWREC: ViewPortType;
   HOR, VERT, RADIUS, KLEUR: Word;

BEGIN
   SetFillStyle(Solidfill,White);
   Bar(0,0,GetMaxX,GetMaxY);
   GetViewSettings(VIEWREC);
   WITH VIEWREC DO
   BEGIN
      HOR  := X2 - X1;
      VERT := Y2 - Y1
   END;
   Randomize;
   FOR RADIUS := 1 TO HOR DIV 2 DO
   BEGIN
      IF RADIUS MOD AFSTAND = 0 THEN
      BEGIN
         REPEAT KLEUR := Random(GetMaxColor)
         UNTIL KLEUR <> White;
         SetColor(KLEUR);
         Circle(HOR DIV 2,VERT DIV 2,RADIUS)
      END;
   END
END;
```

```
PROCEDURE Bochten;
VAR
   VIEWREC  : ViewPortType;
   HOR, VERT: Word;

BEGIN
   SetFillStyle(SolidFill,Cyan);
   SetLineStyle(SolidLn,0,NormWidth);
   Bar(0,0,GetMaxX,GetMaxY);
   GetViewSettings(VIEWREC);
   WITH VIEWREC DO
   BEGIN
      HOR  := X2 - X1;
      VERT := Y2 - Y1
   END;
   SetColor(Black);
   SetFillStyle(SolidFill,Red);
   Arc(HOR * 30 DIV 100,VERT * 20 DIV 100,45,
       270,HOR * 15 DIV 100);
   Arc(HOR * 70 DIV 100,VERT * 20 DIV 100, 270,
       135,HOR * 15 DIV 100);
   Ellipse(HOR DIV 2, VERT DIV 2, 270,90,HOR DIV 4,
           VERT DIV 8);
   Ellipse(HOR DIV 2, VERT DIV 2, 0,360,HOR DIV 2,
           VERT DIV 2);
   FillEllipse(HOR DIV 2, VERT DIV 4*3,HOR DIV 3,
           VERT DIV 9)
END;

PROCEDURE Taart;
CONST
   PERCENTAGES: Array [1..4] of Word = (10,22,37,31);
   START : Word = 0;
   TOTAAL: Word = 0;

   PROCEDURE HaalTekstCoordinaten(MiddenlijnInGraden,
                  RADIUS: Word; VAR X, Y: Integer);
   VAR
      RADIALEN: Real;

   BEGIN
      RADIALEN := MiddenlijnInGraden * Pi / 180;
      X := Round(Cos(RADIALEN) * RADIUS);
      Y := Round(Sin(RADIALEN) * RADIUS)
   END;
```

```
VAR
   VIEWREC: ViewPortType;
   HOR, VERT: Word;
   I, EINDE, KLEUR, LAATSTE_KLEUR: Word;
   PERC_STR: String;
   X, Y: Integer;

BEGIN
   SetFillStyle(SolidFill,LightGreen);
   SetLineStyle(SolidLn,0,NormWidth);
   Bar(0,0,GetMaxX,GetMaxY);
   GetViewSettings(VIEWREC);
   WITH VIEWREC DO
   BEGIN
      HOR  := X2 - X1;
      VERT := Y2 - Y1
   END;
   SetColor(Black);
   KLEUR := Red;
   Randomize;
   FOR I:= 1 TO 4 DO
   BEGIN
     SetFillStyle(SolidFill,KLEUR);
     LAATSTE_KLEUR := KLEUR;
     REPEAT
     KLEUR := Random(GetMaxColor)
     UNTIL NOT(KLEUR IN
     [LAATSTE_KLEUR,Red,LightGreen]);
     Inc(TOTAAL,PERCENTAGES[I]);
     EINDE := TOTAAL * 360 DIV 100;
     PieSlice(HOR DIV 2,VERT DIV 2,START,
              EINDE,HOR DIV 2);
     Str(PERCENTAGES[I],PERC_STR);
     SetTextJustify(CenterText,CenterText);
     IF KLEUR IN [Yellow,White,Lightblue] THEN
     SetColor(Black) ELSE SetColor(White);
     SetTextStyle(SmallFont,HorizDir,4);
     HaalTekstCoordinaten((START + EINDE) DIV 2,
                          HOR DIV 4,X,Y);
     OutTextXY(HOR DIV 2 + X, VERT DIV 2 - Y,
               PERC_STR + '%');
     START := EINDE
   END
END;
```

```
PROCEDURE Tekenen;
VAR
   VIEWREC: ViewPortType;
   HOR, VERT: Word;
   HUIS: Array [0..6] of PointType;

BEGIN
   SetFillStyle(Solidfill,Brown);
   Bar(0,0,GetMaxX,GetMaxY);
   GetViewSettings(VIEWREC);
   WITH VIEWREC DO
   BEGIN
      HOR  := X2 - X1;
      VERT := Y2 - Y1
   END;
   SetColor(Black);
   MoveTo(HOR * 20 DIV 100,VERT * 20 DIV 100);
   LineTo(HOR DIV 2,0);
   LineTo(HOR * 80 DIV 100,VERT * 20 DIV 100);
   LineTo(HOR * 20 DIV 100,VERT * 20 DIV 100);
   SetFillStyle(XHatchFill,Red);
   FloodFill(HOR DIV 2,VERT * 10 DIV 100,GetColor);
   LineTo(HOR * 20 DIV 100,VERT * 40 DIV 100);
   LineTo(HOR * 80 DIV 100,VERT * 40 DIV 100);
   LineTo(HOR * 80 DIV 100,VERT * 20 DIV 100);
   SetFillStyle(LineFill,Red);
   FloodFill(HOR Div 2, VERT * 30 Div 100,GetColor);
   SetFillStyle(XHatchFill,Red);
   HUIS[0].X := HOR * 20 DIV 100;
   HUIS[0].Y := VERT * 70 DIV 100;
   HUIS[1].X := HOR DIV 2;
   HUIS[1].Y := VERT* 50 DIV 100;
   HUIS[2].X := HOR * 80 DIV 100;
   HUIS[2].Y := VERT * 70 DIV 100;
   HUIS[3].X := HOR * 20 DIV 100;
   HUIS[3].Y := VERT * 70 DIV 100;
   FillPoly(4,HUIS);
   SetFillStyle(LineFill,Red);
   Bar(HOR * 20 DIV 100,VERT*70 DIV 100,
       HOR * 80 DIV 100, VERT * 90 DIV 100)
END;

PROCEDURE ZetRijen(HOR,VERT:Word);
CONST
   TELLER    : Word = 0;
   PERCENTAGE: Byte = 15;
```

```
VAR
   I, J: Word;
   X, Y, X1, Y1, X2, Y2, Ruimte: Word;
   Ch: Char;

BEGIN
   X := GetMaxX;
   Y := GetMaxY - (GetMaxY * PERCENTAGE DIV 100);
   FOR I := 0 TO VERT - 1 DO
   BEGIN
      Y1 := (Y DIV VERT * I);
      Y2 := (Y DIV VERT * (I+1));
      Y1 := Y1 + ((Y2-Y1) * PERCENTAGE DIV 100);
      Y2 := Y2 - ((Y2-Y1) * PERCENTAGE DIV 100);
      FOR J := 0 TO HOR - 1 DO
      BEGIN
         X1 := (X DIV HOR * J);
         X2 := (X DIV HOR * (J+1));
         X1 := X1 + ((X2-X1) * PERCENTAGE DIV 100);
         X2 := X2 - ((X2-x1) * PERCENTAGE DIV 100);
         SetViewPort(X1,Y1,X2,Y2,ClipOn);
         CASE TELLER OF
            0: Lijnen;
            1: Rechthoeken;
            2: Bars;
            3: Bar_3D;
            4: Cirkel;
            5: Bochten;
            6: Taart;
            7: Tekenen
         END;
         SetViewPort(0,0,GetMaxX,GetMaxY,True);
         Ruimte := Y DIV VERT * PERCENTAGE DIV 100;
         CASE TELLER OF
            0: BEGIN
                 SetTextStyle(DefaultFont,HorizDir,1);
                 OutTextXY(X1+(X2-X1) DIV 2,
                 Y2+Ruimte,'Lijnen')
               END;

            1: BEGIN
                 SetColor(LightBlue);
                 SetTextStyle(TriplexFont,HorizDir,1);
                 OutTextXY(X1+(X2-X1) DIV 2,
                 Y2+Ruimte,'Rechthoeken vullen')
               END;
```

```
        2: BEGIN
             SetColor(Yellow);
             SetTextStyle(SmallFont ,HorizDir,4);
             OutTextXY(X1+(X2-X1) DIV 2,
             Y2+Ruimte,'Bars met patroon')
           END;

        3: BEGIN
             SetColor(Green);
             SetTextStyle
             (SansSerifFont,HorizDir,1);
             OutTextXY(X1+(X2-X1) DIV 2,Y2+Ruimte,
             '3-D Bars')
           END;

        4: BEGIN
             SetColor(Magenta);
             SetTextStyle( GothicFont,HorizDir,1);
             OutTextXY(X1+(X2-X1) DIV 2,
             Y2+Ruimte,'Cirkels')
           END;

        5: BEGIN
             SetColor(LightCyan);
             SetTextStyle(DefaultFont,HorizDir,1);
             OutTextXY(X1+(X2-X1) DIV 2,
             Y2+Ruimte,'Arc en Ellips')
           END;

        6: BEGIN
             SetColor(Green);
             SetTextStyle(TriplexFont,HorizDir,1);
             SetUserCharSize(1, 2, 1, 3);
             OutTextXY(X1+(X2-X1) DIV 2,
             Y2+Ruimte,'Taart statistiek')
          END;

        7: BEGIN
             SetColor(LightBlue);
             SetTextStyle(SCRIPT_FONT,HorizDir,1);
             OutTextXY(X1+(X2-X1) DIV 2,
             Y2+Ruimte,'LineTo, FillPoly')
          END
      END;
   Inc(TELLER)
END
```

```
  END;
  Ch := ReadKey
END;

VAR
   DRIVER, INSTELLING: Integer;

BEGIN
   DRIVER := Detect;
   InitGraph(DRIVER,INSTELLING,'');
   SetFillStyle(SolidFill,Blue);
   Bar(0,0,GetMaxX,GetMaxY * 5 DIV 100);
   SetTextJustify(CenterText,TopText);
   SetColor(Yellow);
   SetTextStyle(SIMPLEX_FONT,HorizDir,1);
   OutTextXY(GetMaxX DIV 2,0,
      'Demonstratie Borland Graphics InterFace');
   SetTextJustify(CenterText,CenterText);
   ZetRijen(4,2);
   CloseGraph
END.
```

Regels: Toelichting:

3-5Geef font 5 en 6 namen.

[21] **6-32Function BinWord(BitPatroon:String:Word):Word.**

[22] 7-9 Declareer de benodigde lokale variabelen.

[23]11-12Geef B de waarde 0 en J de waarde 1.

[24]13-17 Zet het doorgekregen BitPatroon op een lengte van 16 lettertekens.

[25]18-30Ga de FOR-lus in waar het bitpatroon van het einde tot het begin wordt doorgelopen.

[26]20-24Als er een ander letterteken dan een 1 of een 0 in de string staat, geef BinWord dan de waarde 0 en verlaat de functie.

[27]25-30Vermenigvuldig J met 1 of 0 en tel de uitkomst op bij B. Vermenigvuldig vervolgens J met 2.

31 Geef BinWord de waarde van B.

33-61Procedure Lijnen.

[15]34-35Declareer een getypeerde constante, die gebruikt wordt om de X-positie van de lijn te bepalen.

[16]36-37Declareer een lokale variabele van het type ViewPortType.

[17]39-41Zet de vulkleur op blauw en vul het venster. Zet de tekenkleur op wit.

[18]42 Vraag de coördinaten van het venster op.

[19]45-54Teken met het commando Line een viertal lijnen in het venster en gebruik hierbij telkens een ander voorgedefinieerd lijnpatroon. Voor het veranderen van het patroon wordt SetLineStyle gebruikt.

[20]55-58Zet een lijn waarvan het patroon door de programmeur wordt bepaald.

[28]59 Zet de stijl terug op de standaardstijl.

[29]62-91 Procedure Rechthoeken.

[30]63-66Declareer de benodigde lokale variabelen.

68-69Maak met de GRAPH-procedure Bar een gele rechthoek op het scherm.

[31]70 Zet de coördinaten van het venster in VIEWREC.

71-72 Zet de tekenkleur op blauw en de stijl van de lijn op gewoon.

[32]75-76Bereken de horizontale en verticale maximale waarden.

[33]77-84Maak met behulp van de functie BinWord een eigen patroon om een ruimte te vullen.

[34]85 Stel het in te vullen patroon in op het zojuist gemaakte patroon in de kleur lichtgroen.

[35]86-88Zet met behulp van RecTangle een vierkant in het venster, waarbij aan de buitenzijde 10% ruimte overblijft.

[36]89 Vul het vierkant met het zelfgedefinieerde patroon.

[37] 92-111Procedure Bars.

93-95 Declareer de benodigde lokale variabelen.

97-99 Maak het venster lichtrood en zet de coördinaten in VIEWREC.

102-103Bereken de horizontale en verticale maximale waarden.

[38]104-106Teken een rechthoek, gevuld met het voorgedefinieerde patroon LtSlashFill, in de kleur lichtblauw.

[38]107-109Teken een rechthoek, gevuld met het voorgedefinieerde patroon SlashFill, in de kleur wit.

[39]112-143 Procedure Bar_3D.

[40]113-114Declareer een getypeerde constante om het aantal balken aan te geven dat in het venster moet verschijnen.

[41]115-118 Declareer de benodigde lokale variabelen.

120-122Maak het venster blauw en zet de coördinaten in VIEWREC.

[42]123 Zet het beginpunt voor de Random-functie.

[43]126-127 Bereken de horizontale en verticale maximale waarden.

[44]128-131Bereken de relatieve coördinaten voor de driedimensionale balk.

[45]132-141Zet het aantal balken dat aangegeven staat in de getypeerde constante AANTAL op het scherm. Geef de balken een willekeurige lengte en vul ze met een willekeurig patroon in een willekeurige kleur.

[46]144-170Procedure Cirkel.

[47]145-146Declareer een getypeerde constante om het aantal bits tussen de cirkels aan te geven.

147-149 Declareer de benodigde lokale variabelen.

151-153 Maak het venster wit en zet de coördinaten in VIEWREC.

154-158 Bereken de horizontale en verticale maximale waarden.

[48]159 Zet het beginpunt voor de functie Random.

[49]160-169Ga een FOR-lus in die aangehouden wordt tot de variabele RADIUS gelijk is aan HOR/2.

[50]162-168Als de rest van RADIUS gedeeld door AFSTAND gelijk is aan 0, teken dan een cirkel in een willekeurig gekozen kleur die niet wit is.

[51]171-197Procedure Bochten.

172-174 Declareer de benodigde lokale variabelen.

176-179Maak het venster cyaan en zet de coördinaten in VIEWREC.

180-184Bereken de horizontale en verticale maximale waarden.

185-186 Zet de tekenkleur op zwart en de vulkleur op rood.

[52]187-190 Teken op de aangegeven coördinaten een tweetal bogen.

[53]191-194 Teken op de aangegeven coördinaten een tweetal ellipsen.

[54]195-196Maak met behulp van FillEllipse een ellips die gevuld wordt met het ingestelde patroon.

[55]198-254Procedure Taart.

[56]200-202Declareer een aantal getypeerde constanten.

203-211Lokale procedure HaalTekstCoordinaten.

[66]208-210Reken de X- en Y-waarden uit van een cirkelsegment.

212-217 Declareer lokale variabelen van de procedure Taart.

219-222 Maak het venster lichtgroen en zet de cordinaten in VIEWREC.

223-227Bepaal de horizontale en verticale maximale waarden.

[57]228-229Zet de tekenkleur op zwart en geef de lokale variabele KLEUR de waarde Red.

230 Zet het beginpunt voor de functie Random.

[58]231-253Ga een FOR-lus in voor het verwerken van de getallen die in de getypeerde constante PERCENTAGES staan.

[59]233-234Zet de vulstijl op SolidFill en de vulkleur op KLEUR. Geef LAATSTE_KLEUR de waarde die in KLEUR staat.

[60]235-238Bepaal een contrasterende kleur.

[61]239 Verhoog de lokale variabele TOTAAL met de waarde in de getypeerde constante PERCENTAGES[I].

[62]240 Bereken het aantal graden tot waar een cirkelsegment getekend moet worden.

[63]241-242Teken een cirkelsegment uitgaande van het midden van het venster op de graden aangegeven in START en EINDE. De radius van de cirkel is de helft van het venster.

244 Plaats de tekst gecentreerd rond een X- en Y-punt.

[64]266-246Als het segment gevuld is met geel, wit of lichtblauw, zet dan de kleur op zwart en in andere gevallen op wit.

247 Gebruik SmallFont in de grootte 4 als letter en druk af in horizontale richting.

[65]248-251Roep de lokale procedure HaalTekstCoordinaten aan en druk het percentage af in het cirkelsegment.

[67]252 START wordt gelijk aan EINDE.

[68]255-294Procedure Tekenen.

[69]256-259 Declareer de benodigde lokale variabelen.

261-263Maak het venster bruin en zet de coördinaten in VIEWREC.

264-268Bereken de horizontale en verticale maximale waarden.

[70]269-280Gebruik LineTo om een huisje te tekenen. Vul het dak met het voorgedefinieerde patroon XHatchFill en het huisje met LineFill.

[71]281-290Gebruik FillPoly om met de coördinaten in de array HUIS een dakje te tekenen. Zet het huisje eronder met een aanroep van Bar.

[4]295-386 Procedure ZetRijen.

[5]296-297Declareer de getypeerde constanten TELLER en PERCENTAGE.

[6]299-302 Declareer de benodigde lokale variabelen.

[7]304-305Bereken de waarde van X en Y.
[8]306-384 Ga een FOR-lus in om het aantal gewenste rijen af te werken.
[9]308-311 Bereken de Y1- en Y2-waarden van de rij.
[10]312-383Ga een FOR-lus in om het aantal gewenste horizontale vensters in de rij te maken.
314-317Bepaal de X1- en X2-waarden voor het venster.
[11]318 Roep SetViewPort aan om op de aangegeven coördinaten een venster te maken.
[12]319-328Voer een CASE OF uit met de getypeerde constante TELLER en roep telkens een procedure aan om het venster te vullen.
[13]329 Zet een venster over het hele scherm.
[14]331-381Zet met behulp van CASE OF TELLER een tekst bij het venster. Gebruik hiervoor de beschikbare fonts.

389-402Hoofdprogramma.

[1]390-391 Detecteer de video-adapter en stel de hoogste grafische resolutie in.
[2]392-398 Zet de titel bovenaan in het scherm in een blauwe balk.
[3]399 Zorg dat de tekst geplaatst wordt rond een X- en Y-punt.
400 Roep de procedure ZetRijen aan.
[72]401 Beëindig het gebruik van de unit GRAPH.

Toelichting:

[1]Detect is een in de unit GRAPH gedefinieerde constante voor het automatisch detecteren van het type video-adapter. Deze detectie wordt uitgevoerd door InitGraph als de parameter GraphDriver de waarde Detect heeft. Het resultaat van een dergelijke aanroep is een leeg scherm in de hoogst mogelijke resolutie.

[2]Dit lege scherm krijgt nu eerst aan de bovenkant een blauwe balk met daarin de titel van het scherm. Om dit te bereiken, wordt de kleur en het patroon gekozen waarmee de balk gevuld moet worden. Door aan SetFillStyle de constante SolidFill en de constante Blue mee te geven en daarna de procedure Bar aan te roepen, krijgen we op de opgegeven coördinaten een balk die egaal blauw is.

[3]De aanroep van SetTextJustify zorgt ervoor dat de tekst op een bepaald punt geplaatst wordt ten opzichte van een X- en een Y-coördinaat. Op welke manier dit gebeurt, wordt bepaald door de parameters die aan SetTextJustify worden meegegeven. Hiervoor is in GRAPH weer een aantal constanten gedefinieerd. Deze constanten staan in de tabel op de volgende pagina.

Met SetTextJustify(CenterText,TopText) wordt een instelling gekozen, waarbij CenterText ervoor zorgt dat een X-coördinaat zich in het midden van de tekst bevindt. TopText zorgt ervoor dat de tekst tegen de Y-coördinaat aan geplaatst wordt, waarbij de Y-coördinaat zich boven de tekst bevindt. SetColor stelt de kleur voor de tekst in op geel en met SetTextStyle wordt een font geselecteerd dat in horizontale richting

moet worden afgedrukt. OutTextXY stuurt de tekst ten slotte naar het scherm en geeft de coördinaten voor de plaats van de tekst op. Als met SetTextJustify bepaald is op welke manier tekst op scherm moet worden getoond, dan blijft OutTextXY de teksten op die manier plaatsen. Als de tekst op een andere manier gepresenteerd moet worden, dan moet opnieuw SetTextJustify aangeroepen worden.

Naam:	Waarde:	X/Y:	Betekenis:
LeftText	0	X	Zet de tekst links van de X-coördinaat.
CenterText	1	X	Zet de tekst aan weerszijden van de X-coördinaat.
RightText	2	X	Zet de tekst rechts van de X-coördinaat.
BottomText	0	Y	Zet de tekst op de Y- coördinaat.
CenterText	1	Y	Verdeelt de tekst over de Y-coördinaat.
TopText	2	Y	Zet de tekst onder de Y-coördinaat.

[4]Na het scherm van een titel voorzien te hebben, springen we naar de procedure ZetRijen. In de parameter HOR staat het aantal vensters op de horizontale lijn en in VERT het aantal vensters op de verticale lijn. Deze procedure zet het opgegeven aantal vensters op het scherm. In ieder venster worden één of meer mogelijkheden van de unit GRAPH gedemonstreerd. Ieder venster wordt voorzien van een tekst die steeds in een ander lettertype getoond wordt.

[5]In TELLER wordt bijgehouden hoeveel vensters er zijn gemaakt. In PERCENTAGE wordt de ruimte tussen de vensters bepaald.

[6]De lokale variabelen I en J worden als index in de FOR-lussen gebruikt. De diverse X- en Y-variabelen worden gebruikt voor de positiebepaling van de vensters.

[7]In X en Y wordt het aantal beschikbare horizontale en verticale beeldpunten gezet. GetMaxX levert het aantal beeldpunten op de X-lijn, GetMaxY doet dit voor de Y-lijn. Om aan de boven- en onderkant wat ruimte te houden, wordt Y verlaagd met het percentage dat in de getypeerde constante PERCENTAGE staat.

[8]Daarna gaan we een FOR-lus in. In deze FOR-lus wordt een aantal horizontale rijen van vensters op het scherm gezet. In iedere rij zijn de Y-coördinaten van de vensters hetzelfde.

[9]Hier begint de berekening van de Y-coördinaten die nodig zijn voor het maken van de eerste rij vensters.

[10]Na deze berekening komen we in een geneste FOR-lus waarin voor het aantal vensters dat op de rij gemaakt moet worden, telkens de X-coördinaten uitgerekend worden. Ook hier wordt rekening gehouden met het percentage ruimte tussen de vensters.

Het eerste stel coördinaten dat we verkrijgen, zijn de coördinaten voor het venster links bovenin het scherm.

[11]Deze coördinaten worden gebruikt om met behulp van de GRAPH-procedure SetViewPort op de aangegeven plaats een venster te maken. De linker bovenhoek van een dergelijk venster heeft de coördinaten (0,0). Het coördinaat (0,0) wees eerst naar de linker bovenhoek van het scherm en nu naar de linker bovenhoek van het venster. De X-coördinaten aan de linkerkant van het venster kunnen nu bereikt worden met negatieve coördinaten. De Y-coördinaten boven het venster worden ook met negatieve coördinaten bereikt. MoveTo(-100,-50) beweegt de onzichtbare cursor naar een punt 100 beeldpunten ter linkerzijde en 50 beeldpunten boven het ingestelde venster.

Naast de coördinaten wordt er ook nog een boolean als parameter meegegeven. Deze regelt het zogenaamde clipping. De term "clipping" kan hier vertaald worden met "knippen". Als Clip True aangeeft, dan zullen alle afbeeldingen alleen binnen de opgegeven venstercoördinaten getoond worden. Als afbeeldingen groter zijn dan het gedefinieerde venster, worden als het ware de randen aan de rechter- en onderkant er afgeknipt. Staat Clip op False, dan kunnen er over de coördinaten van het venster heen afbeeldingen gepresenteerd worden. ClipOn en ClipOff zijn constanten die voor dit doel in de unit GRAPH zijn gedefinieerd.

[12]Aan de hand van de waarde in de lokale variabele TELLER wordt bepaald welke procedure aangeroepen moet worden om de faciliteiten van GRAPH te demonstreren.
[13]Bij terugkomst uit de procedure wordt er een venster gemaakt met de de volledige afmetingen van het scherm.

[14]De variabele TELLER wordt door middel van een CASE OF gebruikt om onder het venster een tekst af te drukken. Op deze manier wordt de hele rij vensters afgewerkt. Als de rij klaar is, keren we terug in de lus waar de Y-coördinaten voor de volgende rij berekend worden. Deze procedure gaat door tot alle rijen zijn afgewerkt. Om het programma te volgen, lopen we nu de serie procedures door die in CASE OF aangeroepen worden.

[15]De eerste is de procedure Lijnen. De constante DELER wordt gebruikt om de afstand tussen de te tonen lijnen te berekenen.

[16]De lokale variabele VIEWREC is van het in GRAPH gedefinieerde type ViewPortType en heeft de volgende vorm:

```
ViewPortType = Record
   X1, Y1, X2, Y2: Integer;
   Clip: Boolean;
END;
```

Dit type wordt gebruikt om er de gegevens van het laatst ingestelde venster in te zetten. De lokale variabele PATROON moet straks het patroon bevatten van een door de programmeur gemaakte lijn.

[17]Het venster moet de kleur blauw krijgen. Naast de instelling van patroon en kleur wordt de procedure Bar aangeroepen om van de coördinaat(0,0) tot (MaxX,MaxY) een rechthoek te maken. Deze rechthoek wordt dan gevuld met het ingestelde patroon. GetMaxX en GetMaxY kunnen gebruikt worden, omdat Clip de waarde True heeft en er dus niets buiten het kader van het venster gezet wordt.

[18]Omdat het programma op schermen met verschillende resoluties moet kunnen draaien, moeten de coördinaten relatief bepaald worden. Dat wil zeggen dat er bij het bepalen van de X- en Y-coördinaten rekening gehouden wordt met het aantal beschikbare beeldpunten. De GRAPH-procedure GetViewSettings krijgt als parameter de variabele VIEWREC mee. Bij terugkomst uit de procedure staan in VIEWREC de gegevens van het geldige venster.

[19]De coördinaten uit VIEWREC worden gebruikt om een aantal lijnen verticaal in het venster te zetten. De getypeerde constante DELER wordt hier gebruikt om telkens de X-coördinaten wat op te schuiven. In SetLineStyle wordt het type lijn bepaald, door gebruik te maken van de in GRAPH gedefinieerde constanten.

[20]Als de constante UserBitLn meegegeven wordt, dan moet ook het patroon doorgegeven worden. Dit patroon is van het type Word. Het bitpatroon van de zestien bits die in een Word gaan, bepaalt wat voor patroon er weergegeven wordt. Als alle zestien bits op 1 staan, zal de lijn ononderbroken weergegeven worden. Als een bit op 0 staat, zal het corresponderende beeldpunt afgedrukt worden in de achtergrondkleur. Door de bits te groeperen, onstaat er een patroon.

[21]Om niet telkens een bitpatroon te hoeven uittekenen, hebben we de functie BinWord gemaakt. BinWord moet een string van nullen en enen omzetten naar een numerieke waarde van het type Word. Deze waarde wordt door BinWord geretourneerd en kan gebruikt worden als patroon.

[22]Van de lokale variabelen wordt B gebruikt om de waarde in op te slaan die door BinWord geretourneerd zal worden. I wordt gebruikt als index in een FOR-lus. J is een vermenigvuldigingsfactor in de waardebepaling van een element in de parameter BitPatroon.

[23]De variabele B, die de waarde van het patroon moet gaan bevatten, wordt op 0 gezet en de vermenigvuldigingsfactor J op 1.

[24]Als BitPatroon meer dan zestien lettertekens bevat, dan worden de eerste zestien tekens als patroon gebruikt. Copy(BitPatroon,1,16) hakt zestien tekens van BitPatroon af. Als BitPatroon kleiner is dan zestien tekens, worden er nullen aan het patroon toegevoegd.

[25]De FOR-lus, waar we nu in gaan, loopt BitPatroon door van het laatste naar het eerste teken.

[26]Als een letterteken een andere waarde heeft dan "0" of "1", krijgt de functie de waarde 0 en wordt de functie verlaten.

[27]Het letterteken "0" is het achtenveertigste teken in de ASCII-tabel. De Turbo Pascal- functie Ord('0') retourneert dan ook 48. Het teken "1" neemt de negenenveertigste positie in. Als van de ASCII-waarde van het teken "0" of "1" de waarde 48 afgetrokken wordt, houden we dus de waarde 0 of 1 over. Deze waarde wordt vermenigvuldigd met de vermenigvuldigingsfactor J en opgeteld bij B. Daarna wordt J met 2 vermenigvuldigd om de juiste vermenigvuldigingsfactor te krijgen voor het teken dat aangegeven wordt door BitPatroon[I].

Teruggekeerd in de procedure Lijnen wordt de waarde van BinWord naar SetLineStyle gestuurd.

[28]Ten slotte wordt de stijl weer ingesteld op een gewone lijn met normale dikte. De tekst onder dit venster wordt afgedrukt in het lettertype DefaultFont. Dit lettertype heeft een tekenveld van 8 x 8 bits.

[29]In de procedure Rechthoeken wordt een geel venster met daarin een blauwe rechthoek gemaakt wordt. Deze rechthoek wordt gevuld met een zelfgemaakt patroon.

[30]De lokale variabelen HOR en VERT worden gebruikt om het aantal horizontale en verticale beeldpunten van het venster in vast te leggen. PATROON wordt gebruikt om het zelfgemaakte patroon in vast te leggen.

[31]Na de aanroep van GetViewSettings(VIEWREC) staan in de lokale variabele VIEWREC de gegevens van het venster.

[32]"VIEWREC.X2-VIEWREC.X1" heeft als uitkomst het aantal beschikbare horizontale beeldpunten. Deze uitkomst wordt in de variabele HOR gezet. Het aantal verticale beeldpunten wordt berekend met de formule "VIEWREC.Y2-VIEWREC.Y1". Dit wordt in de variabele VERT gezet.

[33]PATROON is een variabele van het type FillPatternType. Dit type is in GRAPH gedefinieerd en is een Array [1..8] of Byte. De elementen van PATROON hebben elk acht bits. Een patroon kan dus gemaakt worden in de rechthoek van 8 x 8 beeldpunten. Als een bit op 0 staat, wordt het beeldschermpunt in de achtergrondkleur weergegeven, in andere gevallen wordt de kleur gebruikt die met SetFillStyle is ingesteld. De elementen krijgen een waarde door de functie BinWord aan te roepen. Deze functie vertaalt de string met het bitpatroon in een numerieke waarde. In het programma kun je zien dat het patroon op deze manier enigszins zichtbaar wordt.

[34]De GRAPH-procedure SetFillPattern zorgt ervoor dat het zojuist gemaakte patroon ingesteld wordt. Tot er een ander patroon doorgegeven wordt aan SetFillPattern blijft dit patroon bereikbaar met SetFillStyle(UserFill).

[35]RecTangle zet op de opgegeven coördinaten een rechthoek op het scherm in de kleur die met SetColor werd opgegeven. Ook hier zijn de coördinaten weer relatief berekend. De rechthoek laat aan alle zijden 10% van de beeldpunten vrij. Of het nu gaat om een scherm met hoge of met lage resolutie, de rechthoek zal zich steeds op hetzelfde punt in het venster bevinden.

[36]In tegenstelling tot de procedure Bar wordt de ruimte in de rechthoek niet automatisch gevuld met het ingestelde patroon. Hiervoor roepen we de procedure FloodFill aan. FloodFill moet als parameter een X- en een Y-coördinaat meekrijgen, die zich bevinden binnen een door lijnen afgesloten figuur. FloodFill vult dit figuur met het ingestelde patroon. De laatste parameter voor FloodFill is de kleur voor de rand.

De GRAPH-functie GetColor retourneert de met SetColor ingestelde kleurwaarde. De tekst in dit venster wordt afgedrukt als TriplexFont in de kleur lichtblauw.

[37]We zijn al weer aan het derde venster toe en we springen naar de procedure Bars.

[38]Hier wordt een tweetal rechthoeken afgebeeld, die gevuld zijn met een verschillend patroon. De procedure SetFillStyle krijgt hier een tweetal in GRAPH voorgedefinieerde patronen mee, die respectievelijk in lichtblauw en wit worden afgebeeld. De procedure Bar tekent op de opgegeven coördinaten een rechthoek die automatisch gevuld wordt met het ingestelde vulpatroon. De tekst onder dit scherm wordt weergegeven in SansSerifFont in de kleur blauw.

[39]De procedure Bar_3D zal een aantal driedimensionale balken neerzetten tot een willekeurige hoogte, gevuld met een willekeurig patroon in een willekeurige kleur.

[40]De constante AANTAL geeft het aantal balken weer dat moet worden afgebeeld.

[41]Bij de lokale variabelen wordt een aantal variabelen gedeclareerd, voor onder andere de breedte en diepte van de balk.

[42]Na het inkleuren van het venster en het opvragen van de venstercoördinaten wordt Randomize aangeroepen. Randomize zet een beginwaarde voor een tabel van pseudo-willekeurige getallen.

[43]De variabele BREED krijgt als waarde het aantal beeldpunten dat berekend wordt met de formule: "AANTAL horizontale beeldpunten / het gewenste aantal balken+1".

[44]De Y2-coördinaat voor de balk laat aan de onderkant 15% van de beeldpunten vrij. De diepte van een balk wordt bepaald op 1/5 van de breedte. De eerste balk laat aan de linkerkant 10% van

de horizontale beeldpunten vrij.

[45]De FOR-lus wordt doorlopen van 1 tot de waarde in AANTAL. Bij iedere doorloop wordt er een balk getekend. Nu wordt er een willekeurige kleur bepaald. Deze kleur mag niet blauw zijn, want het venster is al blauw gekleurd. De GRAPH-functie GetMaxColor geeft het maximale aantal beschikbare kleuren terug. Random retourneert een willekeurig nummer binnen het bereik 0 tot GetMaxColor. Deze uitkomst wordt doorgegeven aan de procedure SetColor. Omdat de verkregen kleur niet blauw mag zijn, wordt de kleur gekozen in een REPEAT-lus die pas verlaten wordt als de verkregen kleur een andere waarde dan blauw heeft.

Op dezelfde wijze krijgt SetFillStyle zijn parameters door. Er zijn elf voorgedefinieerde patronen. SetFillStyle krijgt binnen het bereik 0 tot 11 een waarde door. Ook hier wordt op willekeurige wijze een kleur gekozen.

De GRAPH-procedure Bar3D tekent op de opgegeven coördinaten een driedimensionale balk, die automatisch gevuld wordt met het opgegeven patroon. De Y1-coördinaat wordt met Random willekeurig bepaald binnen het bereik 0 tot de waarde in VERT. Als Random een waarde toewijst die kleiner is dan Y2, verschijnt de balk onder de lijn die door BODEM wordt aangegeven.

Bij START wordt de inhoud van BREED opgeteld en we keren terug in de lus voor het tekenen van de volgende balk. Als alle balken getekend zijn, komen we weer terug in de procedure ZetRijen waar in groene letters in de stijl SansSerifFont een tekst wordt neergezet.

[46]De procedure Cirkel tekent vanuit het midden van het venster tot aan de rand van het venster een aantal cirkels in willekeurige kleuren.

[47]In de getypeerde constante AFSTAND staat het aantal beeldpunten dat tussen twee cirkels vrij moet worden gelaten.

[48]Voor het willekeurig bepalen van de kleur wordt Randomize aangezet om een beginpunt voor Random te maken.

[49]Hierna wordt een FOR-lus ingegaan die uitgevoerd wordt tot de lokale variabele RADIUS gelijk is aan het aantal horizontale beeldpunten/2. Bij iedere doorloop van de lus wordt RADIUS verhoogd. Als de restwaarde van RADIUS gedeeld door de waarde in AFSTAND gelijk aan 0 is, moet er een cirkel getekend worden. KLEUR krijgt een waarde die niet gelijk is aan White. De kleurwaarde White is namelijk al gebruikt voor de achtergrond. Nadat deze kleur gevonden is, wordt SetColor aangeroepen.

[50]De GRAPH-procedure Circle tekent een cirkel in het venster. Het middelpunt van de cirkel wordt bepaald door de X- en Y-coördinaten. De afmeting van de cirkel wordt bepaald door RADIUS. De tekst wordt eronder gezet in magenta-kleurige Gothische letters.

[51]We gaan nu naar de procedure Bochten. Deze procedure tekent een tweetal cirkelbogen, een onderbroken, een gesloten en een gevulde ellips in het venster.

[52]De GRAPH-procedure Arc tekent cirkelbogen in het venster. Arc krijgt allereerst een X- en een Y-coördinaat mee, die als middelpunt van de cirkel dienen. Dan krijgt Arc het aantal graden op waar de cirkelboog moet starten en vervolgens het aantal graden waar de boog moet eindigen. Nul graden bevindt zich in de cirkel op de stand drie uur. De graden worden daarna tegen de klok in aangegeven. Negentig bevindt zich dan op twaalf uur, honderdtachtig op negen uur, tweehonderdzeventig op zes uur en driehonderdzestig weer op drie uur. De cirkelboog wordt getekend in de richting van START naar EINDE.

[53]Ellipse werkt op precies dezelfde manier als Arc. Uiteraard wordt hier in plaats van een cirkelboog een ellips getekend. Ook de ellips kan onderbroken worden.

De coördinaten van het laatstgegeven Arc- of Ellipse-commando kunnen worden opgevraagd. Je moet dan een variabele definiëren van het type ArcCoordsType. Dit type is in de unit GRAPH gedefinieerd:

```
ArcCoordsType = Record
   X, Y, XStart, YStart, XEnd, YEnd: Integer;
END;
```

Om de coördinaten te krijgen, moet je de GRAPH-procedure GetArcCoords aanroepen. Deze procedure krijgt de variabele van het type ArcCoordsType als parameter mee. Bij terugkeer uit de procedure bevat het record de gezochte coördinaten.

[54]FillEllipse tekent een volledige ellips, uitgaande van de X- en Y-coördinaten. FillEllipse gebruikt hier een X-radius en een Y-radius om de vorm van de ellips te bepalen. Het vult de ellips automatisch met het ingestelde patroon. De tekst wordt eronder gezet in lichtcyaan waarvoor de font DefaultFont gebruikt wordt.

[55]De laatste demo-procedure die we doorlopen, is de procedure Taart. Deze procedure verdeelt een cirkel in segmenten. De kleuren van de verschillende segmenten, die aan elkaar grenzen, mogen niet hetzelfde zijn.

[56]De percentages van ieder deel van de cirkel staan in de getypeerde constante PERCENTAGES. Dit is een array van [1..4] van het type Word. In deze array staan de verhoudingen van de verdelingen van de cirkel. Het totaal van de elementen van PERCENTAGES is 100. De constanten START en TOTAAL worden gebruikt om de graden van een cirkelsegment aan te geven.

[57]De lokale variabele KLEUR krijgt de waarde Red.

[58]We gaan een FOR-lus in die de elementen van PERCENTAGES verwerkt. Met SetFillStyle wordt behalve de waarde van KLEUR, ook aangegeven dat de segmenten egaal gekleurd moeten zijn.

[59]De lokale variabele LAATSTE_KLEUR krijgt de waarde van KLEUR.

[60]Er wordt een willekeurige nieuwe kleur bepaald die niet gelijk is aan LAATSTE_KLEUR, en ook niet aan rood of lichtgroen.

[61]De inhoud van PERCENTAGES[I] wordt opgeteld bij de lokale variabele TOTAAL.

[62]De variabele EINDE krijgt de waarde van de berekening: TOTAAL x 360/100. De lokale variabele EINDE bevat nu het aantal graden waar het segment eindigt.

[63]De GRAPH-procedure PieSlice maakt, uitgaande van het midden van het venster, een gesloten cirkelsegment vanaf het aantal graden aangegeven door START tot het aantal graden aangegeven door EINDE.

Nu zetten we het percentage dat bij dit cirkelsegment hoort midden in het segment. Hiervoor wordt eerst de numerieke waarde die in PERCENTAGES[I] staat, geconverteerd naar een string.

[64]Dan wordt de tekenkleur ingesteld op zwart, mits de kleur van het segment geel, wit of lichtblauw is. Anders wordt de tekenkleur ingesteld op wit. De tekst wordt afgedrukt in SmallFont.

[65]Om nu te berekenen waar het midden van het cirkelsegment is, wordt de lokale procedure HaalTekstCoordinaten aangeroepen. Aan HaalTekstCoordinaten wordt het gemiddelde van het aantal graden in START en in EINDE doorgegeven. Tevens wordt de radius doorgegeven.

[66]De procedure HaalTekstCoordinaten krijgt als VAR-parameters X en Y mee. Om de coördinaten van X en Y uit te rekenen worden de Turbo Pascal-functies Cos en Sin gebruikt. Dit zijn wiskundige functies die op basis van het aantal doorgekregen radialen een X- en een Y-coördinaat berekenen. De uitkomst vermenigvuldigd met de radius resulteert in een tweetal relatieve coördinaten in het cirkelsegment.

Deze berekende coördinaten worden respectievelijk opgeteld en afgetrokken van de X- en Y-coördinaten die het midden van de cirkel aangeven. Deze X- en Y-coördinaten worden meegegeven aan de procedure OutTextXY om PERC_STR in het segment af te drukken.

[67]Nu wordt de waarde van EINDE in START gezet. Bij de volgende doorloop van de lus wordt het nieuwe element van

PERCENTAGES aan TOTAAL toegevoegd, zodat er een nieuwe waarde in EINDE komt te staan.

[68]Als laatste bekijken we de procedure Tekenen. In deze procedure worden twee methoden gebruikt om een figuur te tekenen. In dit geval wordt een huisje met een puntdak getekend.

[69]Bij de lokale variabelen wordt een array van het type PointType gedefinieerd. Dit type is in Turbo Pascal voorgedefinieerd in de volgende vorm:

```
PointType = Record
        X, Y: Integer;
    END;
```

Deze array wordt gebruikt bij het demonstreren van FillPoly.

[70]De eerste manier om een figuur te tekenen is met LineTo. Om LineTo te kunnen gebruiken, gaan we eerst met MoveTo naar het beginpunt van de tekening. LineTo, waaraan een X- en een Y-coördinaat wordt meegegeven, trekt een lijn in de ingestelde stijl naar de opgegeven coördinaten. In eerste instantie wordt het dakje van het huis getekend. Dit is een gesloten driehoek. Het dakje wordt gevuld met een diagonaal ruitjespatroon.

Om het dakje te vullen met het patroon, krijgt FloodFill een X- en een Y-coördinaat mee die zich in de gesloten driehoek bevinden. Daarna wordt op dezelfde manier een rechthoek onder de driehoek gezet. Deze rechthoek wordt met een lijntjespatroon gevuld.

[71]De tweede manier om een figuur te tekenen is het gebruik van FillPoly. Hiermee worden lijnen getrokken met behulp van coördinaten die opgeslagen zijn als data. In dit geval is de variabele HUIS, die een array van het type PointType is, gevuld met de coördinaten die nodig zijn om het dakje te tekenen. FillPoly vult automatisch een omlijnde figuur met het ingestelde patroon. Bij het aanroepen van FillPoly wordt het aantal elementen van de array en de array zelf doorgegeven.

De GRAPH-procedure DrawPoly werkt op dezelfde manier. Deze procedure vult het getekende figuurtje echter niet met een patroon. Het figuurtje wordt afgemaakt door het vulpatroon op streepjes in te stellen en met Bar een rechthoekje te tekenen.

[72]Aan het einde van de demonstratie verlaten we de procedure ZetRijen en komen we terug in het hoofdmenu. Hier wordt met een aanroep van CloseGraph teruggekeerd naar de tekst-mode.

17.10 Scherm op de heap zetten

Het programma GRAPH_3 kopieert een deel van het scherm naar de heap en zet de opgeslagen gegevens weer terug in het scherm. Het programma laat een zon opkomen, die langzaam boven de horizon uitstijgt.

Als je beschikt over een VGA-achtige video-adapter, krijg je in GRAPH_3 tevens gedemonstreerd hoe je de kleuren van het RGB-palet kunt wijzigen. Het RGB-palet is het palet waar de kleurnummers in de verschillende paletingangen naar wijzen.

De kleuren op het scherm moeten gewijzigd worden van zwart in hun uiteindelijke kleur. De zon komt rood op. Als hij bovenaan het scherm is, dan is de kleur langzaam veranderd in geel. De hemel is zwart als de zon opkomt, dan wordt hij paars, en uiteindelijk wordt hij steeds blauwer naarmate de zon verder opkomt. Het gras is eerst zwart en wordt met het stijgen van de zon steeds groener. Het wit van de bloemetjes wordt langzaam zichtbaar:

```
PROGRAM GRAPH_3;
USES CRT, GRAPH;

TYPE
   PWord = ^Word;

CONST
   BR: Byte = 0;  BG:Byte = 0; BB:Byte = 0;
   HR: Byte = 0;  HG:Byte = 0; HB:Byte = 0;
   ZR: Byte = 63; ZG:Byte = 0; ZB:Byte = 0;
   G : Byte = 0;
   HORIZON  = 60;


VAR
   DRIVER, INSTELLING: Integer;
   ZON: Pointer; HOOGTE: PWord;
   ZON_BYTES, DOORSNEE_ZON: Word;
   P: PaletteType;

PROCEDURE MaakZon;
VAR
   X, Y, RADIUS: Word;

BEGIN
   X := GetMaxX DIV 2;
   Y := GetMaxY DIV 2;
   SetLineStyle(SolidLn,0,NormWidth);
   SetColor(Yellow);
   SetFillStyle(SolidFill,Yellow);
   RADIUS := GetMaxX * 20 DIV 200;
   Circle(X,Y,RADIUS);
   FloodFill(X,Y,GetColor);
   SetTextJustify(CenterText,CenterText);
   SetColor(Blue);
   SetTextStyle(SmallFont,HorizDir,4);
   OutTextXY(X,Y,'Turbo Pascal');
   OutTextXY(X,Y+TextHeight('M'),'in de praktijk');
   ZON_BYTES := ImageSize(X-RADIUS-1,Y-RADIUS-1,
                          X+RADIUS+1,Y+RADIUS+1);
   GetMem(ZON,ZON_BYTES);
   GetImage(X-RADIUS-1,Y-RADIUS-1,X+RADIUS+1,
            Y+RADIUS+1,ZON^);
   DOORSNEE_ZON := (RADIUS*2)+2;
   HOOGTE := Ptr(Seg(ZON^),Ofs(ZON^)+2);
   ClearDevice
```

```
END;

PROCEDURE VulScherm;
CONST
   I: Word = 0;
VAR
   SCHEIDING: Word;
   X, Y: Word;

BEGIN
   SetLineStyle(SolidLn,0,NormWidth);
   SetFillStyle(SolidFill,Green);
   ClearDevice;
   SCHEIDING := GetMaxY * HORIZON DIV 100;
   Bar(0, SCHEIDING ,GetMaxX,GetMaxY);
   SetColor(White);
   WHILE I < 500 DO
   BEGIN
      X := Random(GetMaxX);
      REPEAT Y := Random(GetMaxY) UNTIL Y > SCHEIDING;
      PutPixel(X,Y,GetColor);
      Inc(I)
   END
END;

PROCEDURE ZonOp;
VAR
   I: Word;
   X, Y, STIJGING, HEMEL: Word;

BEGIN
   HEMEL := (GetMaxY * HORIZON DIV 100);
   STIJGING := HEMEL DIV 63;
   HOOGTE^ := 0;
   X := (GetMaxX DIV 2)-(DOORSNEE_ZON DIV 2);
   Y := HEMEL - 1;
   FOR I := 0 TO HEMEL DO
   BEGIN
      PutImage(X,Y,ZON^,NormalPut);
      IF I MOD STIJGING = 0 THEN
      BEGIN
         IF (I * STIJGING MOD 2 = 0) AND (BG < 40)
         THEN Inc(BG);
         IF HB < 63 THEN Inc(HB);
         IF HR < 32 THEN Inc(HR) ELSE IF HR > 0
         THEN Dec(HR);
```

```
         IF (ZG < 40) AND (I * STIJGING MOD 2 = 0)
         THEN Inc(ZG);
         IF G < 63 THEN Inc(G)
      END;
      Delay(100);
      IF Y > 0 THEN Dec(Y);
      SetRGBPalette(57,HR,HG,HB);
      SetRGBPalette(2,BR,BG,BB);
      SetRGBPalette(62,ZR,ZG,ZB);
      SetRGBPalette(63,G,G,G);
      IF PWord(HOOGTE)^ < DOORSNEE_ZON THEN
      Inc(HOOGTE^)
   END
END;

BEGIN
   DRIVER := Detect;
   InitGraph(DRIVER,INSTELLING,'');
   GetPalette(P);
   SetBKColor(LightBlue);
   SetRGBPalette(57,HR,BG,HB);
   SetRGBPalette(2,BR,BG,BB);
   SetRGBPalette(62,0,ZG,ZB);
   SetRGBPalette(63,G,G,G);
   SetPalette(8,Black);
   MaakZon;
   VulScherm;
   ZonOp;
   ReadKey;
   SetAllPalette(P);
   FreeMem(ZON,ZON_BYTES);
   CloseGraph
END.
```

Regels: Toelichting:

3-4Definieer een type PWord als pointer naar het type Word.

5-10Definieer een aantal getypeerde constanten, die moeten dienen om het RGB-palet in te stellen. De constante HORIZON wordt gebruikt om de plaats van de horizon in procenten aan te geven.

[7]16-41Procedure MaakZon.

17-18Definieer de benodigde lokale variabelen.

[8]20-21 Laat X en Y naar het midden van het scherm wijzen.

[9]22-23 Stel een gewone lijn in de kleur geel in.

[10]24 Zet het patroon op egaal geel.

[11]25 De zon moet 20% van het totale aantal horizontale beeldpunten beslaan. De RADIUS wordt daarom op 10% gezet (20/200).

[12]26-27Maak een cirkel met de opgegeven coördinaten en vul deze cirkel met het ingestelde patroon.

[13]28-32 Zet een blauwe tekst gecentreerd in de zon.

[14]33-34 Bereken het aantal bytes dat nodig is om de zojuist gemaakte zon als een deel van het scherm op de heap op te slaan.

[15]35 Reserveer het berekende aantal bytes op de heap en zet het adres in de pointervariabele ZON.

[16]36-37Kopieer de bytes, aangegeven door de X- en Y-coördinaten, naar het adres op de heap aangegeven door ZON^.

[17]38 Bereken de doorsnede van het opgeslagen plaatje.

[18]39 Zet in de pointervariabele HOOGTE het adres van het type Word dat de hoogte van het plaatje bevat.

[19]40 Veeg het scherm schoon.

[21]42-62Procedure VulScherm.

43-44Declareer een getypeerde constante voor indexdoeleinden.

45-47Declareer de benodigde lokale variabelen.

[22]49-50Stel een gewone lijn en een egaal groen patroon in.

[23]52 Reken uit op welke hoogte de horizon komt te staan.

[24]53 Zet een groene rechthoek onderaan het scherm tot aan de horizon.

[25]54-60 Zet 500 willekeurige witte beeldpunten in het groene vlak.

[27]63-96 Procedure ZonOp.

[28]68 Bereken het aantal verticale beeldpunten vanaf de horizon tot bovenaan het scherm. Zet de uitkomst in HEMEL.

[29]69 Zet in de variabele STIJGING de uitkomst van HEMEL DIV 63.

```
   [30]70      Zet de hoogte van zon op 0.
   [31]71-72Bereken de X- en Y-coördinaten voor de zon.
   [32]73-95Ga een FOR-lus in waarin I telkens verhoogd wordt. De
          lus duurt tot I de waarde heeft die in HEMEL staat.
   [33]75 Kopieer de bytes aangegeven door ZON^ vanuit de heap
          naar het scherm.
   [34]76-86Als de rest van I/STIJGING de uitkomst 0 heeft, werk
          dan de parameters voor de instelling van het RGB-palet
          bij.
   [35]87      Pauzeer 100 milliseconden.
   [36]88      Verlaag Y zolang Y groter dan 0 is.
   [37]89-92   Werk de verschillende RGB-paletten bij.
   [38]93-94Verhoog het woord aangeduid door het adres in de
          variabele HOOGTE zolang HOOGTE^ kleiner is dan
          DOORSNEE_ZON.
```

97-114Hoofdprogramma.

```
  [1]98-99    Initialiseer de grafische instelling.
   [2]100      Zet het originele palet in de variabele P.
   [3]101      Zet de achtergrondkleur op lichtblauw.
   [4]102-105Werk de RGB-paletten bij.
   [5]106      Zet de kleur zwart op de plaats van lichtblauw.
   [6]107      Roep de procedure MaakZon aan.
   [20]108     Roep de procedure VulScherm aan.
   [26]109          Spring naar de procedure ZonOp.
   [39]110          Wacht op het indrukken van een toets.
   [40]111          Herstel het originele palet.
   [41]112          Geef het geheugen voor ZON vrij.
   [42]113          Sluit de grafische instelling.
```

Toelichting:

[1]De grafische instelling wordt automatisch geplaatst door InitGraph, omdat DRIVER de waarde Detect heeft.

[2]P is een variabele van het type PaletteType. Na een aanroep van GetPalette staat het palet dat op dat moment gebruikt wordt in P.

[3]In het palet krijgt element 0 met SetBKColor de waarde LightBlue.

[4]Bij het aanroepen van SetRGBPalette wordt een viertal parameters meegegeven. De eerste is het kleurnummer. Dit nummer correspondeert met een ingang van de Video-DAC (Digital Analog Converter). De volgende drie parameters geven een numerieke waarde voor rood, groen en blauw door. De Video-DAC werkt met waarden van zes bits. De maximale waarde die doorgegeven kan worden, is dus 63. Als alle drie de parameters op 0 staan, dan wordt zwart weergegeven. Zodra er waarden tussen 0 en 63 worden doorgegeven, ontstaan er kleurnuanceringen. Bij het begin van het programma staan de parameters

voor lichtblauw, groen, geel en wit allemaal op de waarde 0. Dit heeft tot gevolg dat de paletingangen 2, 8, 14 en 15 de kleur zwart in plaats van de oorspronkelijke kleur weergeven.

[5]Omdat we in element 0 van het palet de waarde LightBlue gezet hebben, staat nu zowel in de paletingang 0 als in de paletingang 9 de kleur lichtblauw. Dit is niet nodig. Daarom maken we met behulp van SetPalette van kleur 8 de kleur zwart.

[6]We springen nu naar de procedure MaakZon.

[7]Deze procedure maakt een gele cirkel en plaatst hier een tekst in. Daarna worden de coördinaten van een rechthoek bepaald waarin deze cirkel zich bevindt. Een pointervariabele wijst naar de gegevens. De kleur geel voor de zon is van tevoren zo ingesteld dat die in plaats van geel de kleur zwart weergeeft. Omdat ook de achtergrondkleur zwart is, kun je niet zien dat de zon getekend wordt.

[8]Allereerst wordt het midden van het scherm berekend door het aanroepen van GetMaxX en GetMaxY.

[9]Dan wordt de stijl van de lijn op een doorlopende gele lijn gezet. Hiervoor wordt SetLineStyle aangeroepen.

[10]SetFillStyle zorgt ervoor dat het vulpatroon uit een egaal geel vlak bestaat.

[11]Om een cirkel te kunnen maken, moet een radius, ofwel straal, bepaald worden. Omdat we de cirkel 20% van de beschikbare horizontale beeldpunten willen laten beslaan, is de straal dus 10% van de beeldpunten.

[12]Met de X- en de Y-coördinaat en de straal wordt de GRAPH-procedure Circle aangeroepen. De GRAPH-procedure FloodFill zorgt ervoor dat de cirkel gevuld wordt met een egaal geel vlak. Hiervoor krijgt FloodFill een coördinaat binnen de cirkel mee.

[13]SetTextJustify(CenterText,CenterText) bewerkstelligt dat tekst zodanig op het scherm gepresenteerd wordt, dat de meegegeven coördinaten zich midden in de tekst bevinden. SetTextStyle selecteert een lettertype. In dit geval is gekozen voor SmallFont in horizontale richting met een grootte van 4. OutTextXY stuurt de tekst tenslotte naar het scherm, waar deze geplaatst wordt op de meegegeven coördinaten.

[14]Nu moeten we gegevens uit het videogeheugen gaan kopiëren naar de heap. Als we ruimte op de heap willen gebruiken, zullen we eerst moeten weten hoeveel bytes we moeten reserveren. Dit wordt uitgerekend door de GRAPH-functie ImageSize. Deze functie krijgt als parameters de coördinaten mee van dat deel van het scherm dat we in het geheugen willen zetten. In dit geval zijn dat de X- en Y-coördinaten van de cirkel die respectievelijk verminderd en vermeerderd zijn met de afmeting van de straal. Hierop is een correctie van 1 nodig, omdat we anders precies op de cirkellijn zouden zitten. GetImage rekent uit hoeveel bytes er nodig zijn om de gegevens van de beeldpunten te

kunnen opslaan en telt hier zes bytes bij op. Vier van deze bytes zullen worden gebruikt om de hoogte en de breedte in op te slaan. De waarde van ImageSize wordt in de variabele ZON_BYTES gezet.

[15]ZON_BYTES wordt gebruikt om met GetMem ruimte op de heap te claimen. Het beginadres van deze ruimte wordt in de pointervariabele ZON gezet.

[16]Nu kan de GRAPH-procedure GetImage, met behulp van de coördinaten die ook bij ImageSize gebruikt zijn, de gegevens uit het videogeheugen naar de heap kopiëren.
GetImage zet hierbij in de eerste twee bytes de breedte in beeldpunten en in de volgende twee bytes de hoogte. Aan het einde van de gereserveerde ruimte bevinden zich dan nog twee bytes. Deze bytes zijn gereserveerd.

[17]In de globale variabele DOORSNEE_ZON wordt de doorsnede opgeslagen van het plaatje dat we naar het geheugen gekopieerd hebben.

[18]Omdat we in de procedure ZonOp in het begin maar een deel van de zon nodig hebben, moeten we de hoogte kunnen manipuleren. De hoogte is opgeslagen in byte 2 en 3, gerekend vanaf het adres dat door ZON aangewezen wordt. Door gebruik te maken van de Turbo Pascal-procedure Ptr, waar we op de offset van de pointervariabele ZON twee heb opgeteld, krijgen we in de pointervariabele HOOGTE het adres van de hoogte van het plaatje.

[19]De tekening op het scherm hebben we nu niet meer nodig, deze staat in het geheugen. Met ClearDevice wordt het hele scherm schoongemaakt. Voor het schoonmaken van een venster kunnen we ClearViewPort gebruiken.

[20]We keren terug naar het hoofdprogramma en gaan onmiddellijk naar de procedure VulScherm.

[21]Deze procedure moet aan de onderzijde van het scherm een groen vlak maken, terwijl de bovenkant lichtblauw moet worden. De scheiding van deze twee vlakken wordt aangegeven in de getypeerde constante HORIZON.

[22]Eerst wordt de stijl van de lijn weer ingesteld en het vulpatroon wordt ingesteld op egaal groen.

[23]De lokale variabele SCHEIDING krijgt als waarde de uitkomst van: "het maximaal beschikbare aantal verticale beeldpunten x HORIZON /100".

[24]Dan wordt er met de GRAPH-procedure Bar een rechthoek getekend van de linkerkant naar de rechterkant van het scherm en van het coördinaat aangegeven door SCHEIDING naar de onderkant van het scherm. Bar vult de rechthoek met het ingestelde patroon. Omdat we aan het begin van het programma als achtergrondkleur al lichtblauw gekozen hadden, hoeven we geen lichtblauw vlak meer te tekenen.

[25]Het groene vlak moet een grasveld voorstellen. In een grasveld staan bloemetjes. Dit is een veld vol madeliefjes. Daarom wordt eerst de tekenkleur op wit gezet en worden er met behulp van de Random-functie coördinaten in het grasveld bepaald. PutPixel kleurt vervolgens het aangewezen beeldpunt met de tekenkleur. Naast PutPixel heeft de unit GRAPH ook nog de functie GetPixel(X,Y). Deze functie retourneert het kleurnummer van het aangewezen beeldpunt.

[26]We keren opnieuw terug in het hoofdprogramma. We hebben nu een zon op de heap geplaatst en het scherm in vlakken verdeeld. Hiervan is echter niets zichtbaar, omdat de gebruikte kleuren nog steeds als zwart weergegeven worden.

[27]De procedure ZonOp maakt deze kleuren zichtbaar. De procedure begint met een donker scherm. De zon moet nu geleidelijk aan boven de horizon verschijnen en tegelijkertijd, gaandeweg de stijging, van rood naar geel verkleuren. Als de zon opkomt, is de hemel eerst nog rood/paars, maar naarmate de zon geler wordt, wordt de hemel blauwer. Ook het groen van het gras is eerst niet zichtbaar. Tijdens het opkomen van de zon wordt dit steeds helderder. Dit geldt ook voor de witte bloemetjes in het gras.

[28]De lokale variabele HEMEL krijgt als waarde het aantal verticale beeldpunten, gerekend vanaf de bovenkant van het scherm tot aan de horizon.

[29]In de lokale variabele STIJGING wordt de uitkomst van HEMEL/63 gezet. De RGB-paletten kunnen maximaal de waarde 63 krijgen. Dit zijn zes bits. In STIJGING staat nu om de hoeveel beeldpunten de paletten bijgewerkt moeten worden om een geleidelijke aanpassing van de kleuren te krijgen.

[30]De zon moet langzaam opkomen. We hebben daarom in het begin maar een klein randje van het opgeslagen plaatje nodig. In de procedure MaakZon heeft de globale pointervariabele HOOGTE het adres gekregen van de bytes waar de hoogte van het plaatje opgeslagen is. HOOGTE is een variabele van het type PWord. Dit type is aan het begin van het programma gedefinieerd als een pointer naar het type Word. Op het moment dat we de zon gaan laten opkomen, zetten we HOOGTE^ op 0.

[31]Om de zon in het midden van het scherm op te laten komen, zetten we in de X-coördinaat eerst het midden van de horizontale lijn en trekken daar de helft van de doorsnede van het plaatje van af. De Y-coördinaat-1 is de waarde die in HEMEL staat. De correctie is nodig om de zon net boven de horizon te laten

opkomen.

[32]We gaan nu een FOR-lus in die bij iedere doorloop I met 1 verhoogt, net zolang tot de waarde in HEMEL bereikt is.

[33]De GRAPH-procedure PutImage kopieert de beeldschermgegevens naar de opgegeven coördinaten op het scherm. Het adres van deze gekopieerde gegevens wordt aangegeven door ZON^. De derde parameter bepaalt hoe de deze gegevens op het scherm zullen verschijnen.

Naam:	Waarde:	Bewerking:
NormalPut	0	MOV
CopyPut	0	MOV
XORPut	1	XOR
ORPut	2	OR
ANDPut	3	AND
NOTPut	4	NOT

NormalPut en CopyPut maken het gekopieerde plaatje zichtbaar. De andere constanten voeren eerst de logische bewerkingen op de bits uit. XORPut maakt bijvoorbeeld een plaatje beurtelings zichtbaar en onzichtbaar. Zie voor de bewerkingen ook het hoofdstuk over logische operatoren.

Deze constanten worden niet alleen gebruikt bij de GRAPH-procedure PutImage, maar ook bij de procedure SetWriteMode. Bij SetWriteMode worden de bewerkingen uitgevoerd op lijnen. Zo kan een lijn onzichtbaar gemaakt worden door SetWriteMode aan te roepen en als parameter XORPut mee te geven.

[34]Als de restwaarde van I/STIJGING op 0 uitkomt moet het RGB-palet bijgewerkt worden. Het groen van de bodem, waarvan de groencomponent aangegeven wordt door de variabele HG, wordt langzaam groener omdat HG in kleine stapjes verhoogd wordt tot de waarde 40 is bereikt. In HR staat de waarde van de roodcomponent, die in de kleur lichtblauw voor de hemel gebruikt wordt. Deze stijgt eerst naar 32, om daarna naar 0 te zakken. Hierdoor krijgen we eerst toenemend rood in de hemel, dat daarna wegtrekt.

[35]Als de variabelen voor de kleuren gezet zijn om doorgegeven te kunnen worden, dan wordt de uitvoering van het programma even onderbroken. Hiervoor gebruiken we de Turbo Pascal-procedure Delay. De waarde die Delay meekrijgt, is het aantal milliseconden dat de uitvoering onderbroken wordt. Als we niet zouden pauzeren, zou de zon veel te vlug stijgen.

[36]De Y-coördinaat moet langzaam in waarde verminderen tot de waarde 0 bereikt is. In dat geval staat de zon hoog aan de hemel.

[37]De kleurwaarden die we voor de verschillende kleuren berekend hebben, wordt met SetRGBPalette in de Video DAC-register gezet.

[38]De hoogte van het af te beelden plaatje wordt verhoogd tot de maximale hoogte bereikt is. Deze maximale hoogte staat in DOORSNEE_ZON.

[39]Als de zon zijn reis volbracht heeft, keren we terug in het hoofdprogramma waar ReadKey ervoor zorgt dat de uitvoering van het programma gestopt wordt zodra er een toets wordt ingedrukt.

[40]Nadat de toets is ingedrukt, wordt het programma beëindigd. Het palet wordt weer in zijn oorspronkelijke staat hersteld. Het oude palet dat was opgeslagen in de variabele PSetAllPalette(P) maakt het palet dat in P staat weer het geldende palet. In plaats van deze omslachtige methode die terwille van de demonstratie gekozen is, kun je ook gewoon zeggen: SetAllPalette(GetDefaultPalette). De GRAPH-functie GetDefaultPalette retourneert het oorspronkelijke palet. De procedure GraphDefaults gaat nog een stapje verder. Deze procedure zet CP (de onzichtbare cursor) op het coördinaat (0,0). Verder zet deze procedure alle oorspronkelijke waarden terug, inclusief het palet.

[41]Voor de pointer ZON hadden we ruimte op de heap geclaimd. Deze ruimte wordt weer vrijgegeven.

[42]CloseGraph geeft tenslotte de ruimte die GRAPH op de heap had vrij en herstelt de toestand weer zoals deze was voor InitGraph werd aangeroepen.

17.11 Opslag op schijf

De GRAPH-procedures GetImage en PutImage worden ook gebruikt bij het opslaan van grafische schermen op de schijf. Deze procedures worden gebruikt in combinatie met BlockRead en BlockWrite. Deze laatste Turbo Pascal-commando's is uitgebreid behandeld in het hoofdstuk over bestanden. In de vorige paragraaf hebben we de gegevens voor een deel van het scherm naar de heap gekopieerd. Om te bepalen hoeveel bytes hiervoor nodig waren, werd ImageSize gebruikt. Als het aantal bytes groter is dan 64 Kb, retourneert GetImageSize een 0. Er zijn video-adapters die instellingen kennen die groter zijn dan 64 kilobyte. De oplossing is dan om het scherm in delen naar het bestand te schrijven. GRAPH_4 laat zien hoe dat gedaan wordt.

In het programma GRAPH_4 kun je zien hoe de GRAPH-procedure LineRel gebruikt wordt. Deze procedure trekt lijnen vanaf een relatief punt. Er wordt eerst een kubus getekend met behulp van Bar3D, vervolgens wordt dezelfde kubus vervaardigd met behulp van LineRel. Tenslotte laat GRAPH_4 zien op welke manier de GRAPH-procedure GetAspectRatio gebruikt wordt.

We hebben gezien hoe in de loop van de tijd bij de verschillende video-adapters het aantal beeldpunten in verticale richting groter werd. In de lagere resoluties is een beeldpunt langer dan hij breed is. Het gevolg hiervan is dat, als je volgens de opgegeven coördinaten een vierkant tekent, dit vierkant als een rechthoek op het scherm verschijnt. Hierdoor ziet een cirkel er

`uit als een ellips.` GetAspectRatio compenseert dit.

```
PROGRAM GRAPH_4;
USES CRT, GRAPH;

VAR
   DRIVER, INSTELLING: Integer;

FUNCTION TekenFiguren: Char;
VAR
   BREED, HOOG, DIEP, ASPECT_X, ASPECT_Y: Word;
   X, Y: Integer;
   I: Byte;

   PROCEDURE VulVlakken(B,H:Word;T:Boolean);
   VAR
      X, Y: Integer;

   BEGIN
      X := GetX;
      Y := GetY;
      SetFillStyle(SolidFill,Red);
      FloodFill(X+1,Y+1,GetColor);
      SetFillStyle(SolidFill,Green);
      FloodFill(X+B+1,Y+H DIV 2,GetColor);
      IF T THEN
      BEGIN
         SetFillStyle(SolidFill,Yellow);
         FloodFill(X+B DIV 2,Y-1,GetColor)
      END
   END;


BEGIN
   X := GetMaxX DIV 20;
   Y := GetMaxY DIV 10;
   BREED := GetMaxX DIV 5;
   HOOG  := BREED;
   DIEP  := BREED DIV 4;
   Bar3D(X,Y,X+BREED,Y+HOOG,DIEP,TopOn);
   MoveTo(X,Y);
   VulVlakken(BREED,HOOG,True);
   X := X + BREED + DIEP + (GetMaxX DIV 10);
   MoveTo(0,0);
   FOR I := 0 TO 1 DO
   BEGIN
      MoveTo(X,Y);
      LineRel(0,HOOG);
```

```
      LineRel(BREED,0);
      LineRel(0,-HOOG);
      LineRel(-BREED,0);
      LineRel(BREED DIV 4,-DIEP * 3 DIV 5);
      LineRel(BREED,0);
      LineRel(0,HOOG);
      LineRel(-BREED DIV 4,DIEP * 3 DIV 5);
      MoveRel(0,-HOOG);
      LineRel(BREED DIV 4,-DIEP * 3 DIV 5);
      MoveTo(X,Y);
      VulVlakken(BREED,HOOG,True);
      GetAspectRatio(ASPECT_X,ASPECT_Y);
      HOOG := Round(ASPECT_X / ASPECT_Y * BREED);
      X := X + BREED + DIEP + (GetMaxX DIV 10)
   END;
   SetTextStyle(SmallFont,HorizDir,6);
   Y := GetMaxY - (5*TextHeight('M'));
   SetWriteMode(XORPut);
   SetTextJustify(LeftText,CenterText);
   OutTextXY(GetMaxX DIV 2,Y,
            'Wilt U het scherm opslaan? (J/N)');
   TekenFiguren := ReadKey;
   OutTextXY(GetMaxX DIV 2,Y,
            'Wilt U het scherm opslaan? (J/N)');
END;

PROCEDURE SlaSchermOp;
CONST
   VAN: Longint = 0;


VAR
   DELEN: Array [1..4] of Word;
   AANTAL_BYTES: Word;
   I: Byte;
   F: File;
   P: Pointer;

BEGIN
   Assign(F,'PLAATJE.GRF');
   Rewrite(F,1);
   FOR I := 1 TO 3 DO DELEN[I] := (GetMaxY DIV 4) * I;
   DELEN[4] := GetMaxY;
   FOR I := 1 TO 4 DO
   BEGIN
      AANTAL_BYTES := ImageSize(0,VAN,GetMaxX,DELEN[I]);
```

```
      GetMem(P,AANTAL_BYTES);
      GetImage(0,VAN,GetMaxX,DELEN[I],P^);
      BlockWrite(F,P^,AANTAL_BYTES);
      FreeMem(P,AANTAL_BYTES);
      VAN := DELEN[I]
   END;
   RestoreCrtMode;
   ClrScr;
   GotoXY(20,10);
   Write
   (’Het grafische scherm is opgeslagen op schijf’);
   GotoXY(20,11);
   Write(’      We zijn nu weer in een tekstscherm’);
   GotoXY(20,12);
   Write
   (’  Druk op een toets om het scherm weer in te lezen’);
   ReadKey;
   DetectGraph(DRIVER,INSTELLING);
   SetGraphMode(1);
   Reset(F,1);
   VAN = 0;
   FOR I := 1 TO 4 DO
   BEGIN
      AANTAL_BYTES := ImageSize(0,VAN,GetMaxX,DELEN[I]);
      GetMem(P,AANTAL_BYTES);
      BlockRead(F,P^,AANTAL_BYTES);
      PutImage(0,VAN,P^,NormalPut);
      FreeMem(P,AANTAL_BYTES);
      VAN := DELEN[I]
   END;

   OutTextXY(GetMaxX DIV 2,
   GetMaxY - (5*TextHeight(’M’)),
   ’Druk op een willekeurige toets...’);
   ReadKey
END;

BEGIN
   DRIVER := Detect;
   InitGraph(DRIVER,INSTELLING,’’);
   SetGraphMode(1);
   SetBKColor(LightGray);
   IF TekenFiguren IN [’J’,’j’] THEN SlaSchermOp;
   CloseGraph
END.
```

Regels: Toelichting:

2 Gebruik de units CRT en GRAPH.
3-4Declareer de benodigde globale variabelen.
5-65Function TekenFiguren.
6-9Declareer de benodigde lokale variabelen.
[6]10-25Lokale procedure van TekenFiguren.
Procedure VulVlakken(B,H:Word; T:Boolean).
11-12Declareer lokale variabelen van VulVlakken.
[7]14-15Bepaal de positie van de onzichtbare cursor en zet deze in X en Y.
[8]16-24Geef de verschillende vlakken van de kubus een kleur.
[27]24-31Bepaal de coördinaten voor het tekenen van de eerste kubus.
[5]32-33Teken met Bar3D een kubus op het scherm en zet de onzichtbare cursor op het punt (X1, Y1) van de kubus.
[6]34Roep de lokale procedure VulVlakken aan om de vlakken van de kubus in te kleuren.
35 Reken de X-coördinaat uit voor de tweede kubus.
36-55Ga een FOR-lus in die twee keer doorlopen wordt om twee kubussen te tekenen.
39 Plaats de cursor op het beginpunt.
[9]40-49Teken een kubus.
50-51 Ga naar het oorspronkelijke beginpunt en roep de procedure VulVlakken aan.
[10]52-53Roep GetAspectRatio aan en compenseer de variabele HOOG met de gevonden uitkomst.
54 Reken de X-coördinaat uit voor de nieuwe kubus en ga terug naar het begin van de FOR-lus om de nieuwe kubus te tekenen met een gecompenseerde variabele HOOG.
[11]56-62Zet tekst op het scherm en geef de waarde van de ingedrukte toets aan de functie TekenFiguren.
66-117Procedure SlaSchermOp.
67-74Declareer een lokale constante en de benodigde variabelen.
[12]76-77Maak een bestand aan.
[13]78-79Deel het scherm in vieren en zet de uitkomsten in de array DELEN.
[14]80-88Ga een FOR-lus in die vier keer doorlopen wordt om bij iedere doorgang een deel van het scherm in het bestand te schrijven.
[15]89-99Ga terug naar een tekstscherm en meldt dat het grafische scherm op schijf gezet is.
[16]100-101Ga terug naar de grafische instelling en stel de juiste mode in.
[17]102 Zet de bestandswijzer aan het begin van het

bestand.
103 Zet in de lokale variabele VAN het startpunt voor het lezen.
[18]104-112Lees het scherm deel voor deel van schijf en kopieer het naar de videobuffer.
[19]113-116 Zet tekst op het scherm en wacht op het indrukken van een toets.

118-125 Hoofdprogramma.

[1]119-121Stel de grafische instelling in.
[2]122 Maak het scherm egaal lichtgrijs.
[3]123 Als de functie TekenFiguren een "J" of een "j" retourneert, roep dan de procedure SlaSchermOp aan.
[20]124 Sluit de grafische instelling.

Toelichting:

[1]Bij het opstarten van het programma GRAPH_4 wordt een lage resolutie ingesteld.

[2]De achtergrondkleur van het scherm wordt op lichtgrijs gezet.

[3]We roepen nu eerst TekenFiguren aan. Deze functie tekent drie kubussen op het scherm. De laatste kubus heeft voor de hoogte een compensatie gekregen.

[4]Als we TekenFiguren betreden, moeten we eerst bepalen waar het beginpunt voor de eerste kubus komt. We laten 5% van het aantal beeldpunten aan de linkerkant vrij. Aan de bovenkant blijft 10% vrij. De breedte van de kubus stellen we in op 20% van het beschikbare aantal beeldpunten op de horizontale lijn. De hoogte wordt gelijk aan de breedte, en de diepte wordt op 25% van de breedte bepaald.

[5]Met deze coördinaten wordt de procedure Bar3D aangeroepen. De voorgedefinieerde variabele TopOn van het type Boolean geeft aan dat ook het bovenaanzicht van de kubus getekend moet worden. Naast TopOn is er ook nog een variabele TopOff. Als deze wordt meegezonden, wordt het bovenaanzicht niet getekend. De onzichtbare cursor wordt gepositioneerd op de X1,Y1-hoek van de kubus.

[6]Nadat Bar3D een kubus getekend heeft, wordt de lokale procedure VulVlakken aangeroepen.

[7]De GRAPH-procedures GetX en GetY retourneren de positie van de cursor.

[8]De gevonden X- en Y-waarden worden aangewend om een punt in een vlak te berekenen. Hiervoor worden ook de parameters B en H

gebruikt die de breedte en de hoogte bevatten. Voordat de procedure FloodFill aangeroepen wordt, stellen we een vulpatroon in. FloodFill wordt nu drie keer aangeroepen met de berekende coördinaten en vult telkens een vlak van de kubus met het ingestelde patroon.

[9]Nadat Bar3D de eerste kubus geproduceerd heeft, gaan we met behulp van LineRel en MoveRel zelf een kubus tekenen. MoveRel beweegt de onzichtbare cursor naar een punt ten opzichte van de plaats waar de cursor zich bevond. LineRel tekent een lijn vanaf de plaats waar de cursor zich bevindt naar de opgegeven plaats. De plaats waar de cursor zich bevindt, wordt door LineRel gezien als (0,0). Als we een lijn van 200 beeldpunten in de breedte willen trekken vanaf het punt waar de cursor staat, zetten we: "LineRel(200,0)". Er wordt nu een lijn getrokken van 200 beeldpunten lang, beginnend rechts van de cursor. Omdat de doorgegeven Y-parameter een 0 was blijft de lijn op de Y-coördinaat waar de cursor zich bevond. Willen we een lijn links van de cusor trekken, dan wordt een negatieve coördinaat doorgegeven: LineRel(-200,0).

[10]Voordat de derde kubus getekend kan worden, gaan we de hoogte compenseren. In de lage resolutie zullen de kubussen niet vierkant zijn, maar in de lengte uitgerekt. Toch hadden we de breedte en de hoogte van dezelfde waarde voorzien. Dit verschijnsel wordt veroorzaakt door het feit, dat in de lage resolutie de lengte van een beeldpunt groter is dan de breedte. Het gevolg is dat de onderlinge maten van een tekening op het scherm niet kloppen.

GetAspectRatio krijgt als parameter een tweetal integers mee. Bij terugkeer uit de procedure bevatten deze parameters de verhouding van X ten opzichte van Y. In de EGA-mode 640 x 350 heeft ASPECT_X de waarde 7750 en ASPECT_Y de waarde 10000. We hadden gezegd dat HOOG gelijk werd aan BREED. Daarmee krijgen we echter geen vierkant maar een rechthoek. HOOG moet dus kleiner worden. Als ASPECT_X gedeeld wordt door ASPECT_Y, dan is de uitkomst 0.775. Dit is het getal waarmee BREED vermenigvuldigd moet worden om de juiste verhouding in HOOG te kunnen zetten. BREED kreeg in deze mode de waarde 128. Dit was de waarde die ook in HOOG stond. Na de compensatie met GetAspectRatio heeft HOOG de waarde 98. Als er nu teruggekeerd wordt in de lus en er opnieuw een kubus getekend wordt, zal deze kubus even hoog als breed zijn.

[11]Als de kubussen getekend zijn, willen we weten of ze opgeslagen moeten worden op schijf. Eerst wordt er een lettertype gekozen, in dit geval SmallFont. Dan wordt de Y-coördinaat voor de tekst bepaald. Hiervoor wordt de GRAPH-functie TextHeight gebruikt. Als parameter wordt de letter M meegegeven. Dit is de hoogste letter. De GRAPH-procedure SetWriteMode stelt de manier in waarop de tekst wordt getoond. In dit geval werd de voorgedefinieerde variabele XORPut als parameter doorgezonden. Met deze instelling zal de tekst, als ik die voor een tweede keer naar het scherm stuur, verdwijnen en wordt het oorspronkelijke patroon hersteld.

Deze manier werkt uitsluitend bij de instelbare fonts. Het DefaultFont, met een tekenveld van 8 x 8 bits, kan niet op deze manier behandeld worden. Met ReadKey krijgt de functie TekenFiguren nu de waarde van de ingedrukte toets. Voor we de

functie verlaten, sturen we nog even de tekst naar het scherm. Hierdoor verdwijnt de tekst en wordt er dus een scherm met alleen kubussen opgeslagen.

We komen terug in het hoofdprogramma. Hier wordt gekeken of MaakFiguren een "J" of een "j" retourneert. Is dit het geval, dan wil de gebruiker het scherm dus opslaan en roepen we SlaSchermOp aan.

[12]Er wordt eerst een bestand PLAATJE.GRF aangemaakt. Rewrite zet dit bestand op het scherm. Door bij de Rewrite-opdracht een grootte van 1 door te geven, is het bestand geschikt om er enkele bytes in te schrijven.

[13]Als gezegd kan GetImageSize maximaal 64 Kb bevatten. Menige grafische instelling gebruikt meer bytes. Dit probleem kan worden opgelost door het scherm in delen weg te schrijven. Hiervoor gebruiken we de lokale variabele DELEN, een Array [1..4] of Word. In de elementen van DELEN worden de coördinaten van Y berekend.

Omdat er bij een deling een restwaarde kan overblijven, wordt in het laatste element de waarde van GetMaxY geschreven. De X-coördinaten zullen, omdat het om het hele scherm gaat, altijd van 0 tot GetMaxX lopen.

[14]In deze volgende FOR-lus wordt met GetImageSize het benodigde aantal bytes opgehaald om het schermdeel eerst naar het geheugen te kopiëren, en vervolgens naar schijf. In de variabele VAN staat de eerste Y-coördinaat, en in DELEN[I] de tweede. GetMem zorgt ervoor dat P naar het aantal benodigde bytes op de heap wijst. BlockWrite kopieert daarna het aantal bytes vanuit de heap naar het geopende bestand. FreeMem geeft het geheugen weer vrij en de variabele VAN krijgt de waarde in DELEN[I]. Als we terugkeren in de lus, wordt het tweede deel van het scherm verwerkt. Dit gaat zo door tot het hele scherm verwerkt is.

[15]De GRAPH-procedure RestoreCrtMode brengt ons terug in een tekstscherm. Deze procedure laat de grafische gegevens, die door de procedure InitGraph op de heap gezet zijn, onaangetast. In het tekstscherm verschijnt de mededeling dat we nu weer in een tekstscherm zijn en dat na een druk op een toets het grafische scherm weer ingelezen zal worden.

[16]DetectGraph brengt ons nu weer terug in de grafische instelling. Als hier InitGraph gebruikt zou zijn, zou er een tweede gegevensset op de heap zijn gezet. DetectGraph doet dit niet. De oorspronkelijke gegevensset wordt gewoon weer gebruikt. De grafische mode wordt weer net zo ingesteld als voordat we het grafische scherm wegschreven.

[17]De bestandswijzer wordt met behulp van Reset teruggezet naar

het begin van het bestand.

[18]Omdat de array DELEN nog steeds waarden bevat, gebruiken we deze om de delen van het scherm weer in te lezen. Het lezen wordt op dezelfde manier gedaan als het schrijven. Daar werden de functies GetImage en BlockWrite gebruikt. Hier worden de functies PutImage en BlockRead gebruikt.

[19]Onderaan het scherm verschijnt de tekst "Druk op een willekeurige toets...". ReadKey onderbreekt de uitvoering tot er een toets wordt ingedrukt.

[20]CloseGraph vernietigt de gegevens die door InitGraph op de heap gezet zijn.

17.12 Grafisch scherm naar de printer sturen

Turbo Pascal kent geen mogelijkheden om grafische afbeeldingen naar de printer te sturen. Toch is het wel aardig om een afbeelding geheel of gedeeltelijk af te kunnen drukken. De unit GRPRINT.PAS kan afbeeldingen zowel naar een printer sturen die met de Hewlett Packard's PCL-opdrachtenset werkt, als naar een matrixprinter die in de Epson-mode staat.

De unit is ontworpen als een zogenaamde Text File Device Driver. Dit wordt ook wel een TFDD genoemd. Het betreft hier een door Turbo Pascal geboden techniek om eigen functies aan in- en uitvoerroutines te koppelen. In het hoofdstuk over bestanden staat vermeld dat Turbo Pascal apparaten als een file beschouwt. De printer wordt in de unit gedefinieerd als een file van het type Text. Iedere file van het type Text beschikt over een gegevensblok. Deze gegevens zijn opgeslagen in de vorm van het type TextRec. Dit type wordt gedefinieerd in de unit DOS en is al behandeld in hoofdstuk 8, "Bestanden":

```
TextRec = Record
   Handle   : Word;
   Mode     : Word;
   BufSize  : Word;
   Private  : Word;
   BufPos   : Word;
   BufEnd   : Word;
   BufPtr   : ^TextBuf;
   OpenFunc : Pointer;
   InOutFunc: Pointer;
   FlushFunc: Pointer;
   CloseFunc: Pointer;
   UserData : Array [1..16] of Char;
   Name     : Array [0..79] of Char;
   Buffer   : TextBuf
```

```
END;
```

De buffer is gedeclareerd als type TextBuf. Dit type is in de DOS-unit gedefinieerd:

TextBuf = Array [0..127] of Char;

Dit betekent dat iedere file van het type Text de beschikking heeft over een standaardbuffer van 128 bytes. Door de juiste velden te veranderen, kan de textfile aan een grotere buffer gekoppeld worden. De pointers OpenFunc tot en met CloseFunc zijn bestemd om de adressen te bevatten van de zelfgeschreven functies. De unit GRPRINT.PAS laat zien hoe dat gaat. Om deze unit uit te testen, kun je in het programma GRAPH_4.PAS in de USES_regel "GRPRINT" opnemen. Voordat het programma beëindigd wordt, kun je nu opnemen "Drukvenster(2)". Als je een Epson-printer gebruikt, zet dan voor je DrukVenster aanroept "PrinterType := Epson". Een $-teken is de aanduiding van een hexadecimaal getal:

UNIT GRPRINT;

```
Interface

USES CRT, DOS, GRAPH;

TYPE
   PrinterTypen = (HP,EPSON);

CONST
   PRINTERTYPE: PrinterTypen = HP;
VAR
   PRN: Text;

PROCEDURE DrukVenster(Dichtheid:Word);
Implementation

CONST
   ESC = #27;
VAR
   BREEDTE, HOOGTE: Word;
   VIEWPORT: ViewPortType;
   INTRO   : String[10];

{$F+}
FUNCTION NulFunctie(Rec:TextRec):Integer;
{$F-}

BEGIN
   NulFunctie := 0;
END;

{$F+}
FUNCTION UitvoerNaarPrinter(VAR Rec:TextRec):Integer;
{$F-}

VAR
   REG: Registers;
   WIJZER: Word;

BEGIN
   WITH REC DO
   BEGIN
      WIJZER := 0;
      REG.AH := 16;
      WHILE (WIJZER<BufPos) AND (REG.AH AND 16 = 16) DO
      BEGIN
```

```
         REG.AH := 0;
         REG.AL := Ord(BufPtr^[WIJZER]);
         REG.DX := UserData[1];
         Intr($17,REG);
         INC(WIJZER)
      END;
      BufPos := 0;
      IF REG.AH AND 16 = 16 THEN
      UitvoerNaarPrinter := 0
      ELSE
      IF REG.AH AND 32 = 31 THEN
      UitvoerNaarPrinter := 159
      ELSE
      UitvoerNaarPrinter := 160
    END
END;

PROCEDURE InitHP(Dichtheid:Integer);
CONST
   CURSORPOSITIE: String = '5';

VAR
   PUNTEN_PER_INCH: String[3];

BEGIN
   CASE Dichtheid OF
      1: PUNTEN_PER_INCH := '75';
      2: PUNTEN_PER_INCH := '100';
      3: PUNTEN_PER_INCH := '150';
      4: PUNTEN_PER_INCH := '300';
   ELSE  PUNTEN_PER_INCH := '100'
   END;
   Write(PRN, ESC + 'E');
   Write(PRN, ESC + '*t' + PUNTEN_PER_INCH + 'R');
   Write(PRN, ESC+ '&a' + CURSORPOSITIE + 'C');
   Write(PRN, ESC + '*r1A')
END;

PROCEDURE InitEpson(Dichtheid:Integer);
VAR
   REGELAFSTAND:String[10];

BEGIN
   REGELAFSTAND := #27 + '3' + #24;
   CASE Dichtheid OF
      1:INTRO := #27 + 'K';
```

```
      2:INTRO := #27 + 'L';
      3:INTRO := #27 + 'Y';
      4:INTRO := #27 + 'Z';
   ELSE INTRO := #27 + 'L'
   END;
   Write(PRN, REGELAFSTAND)
END;

PROCEDURE SluitHP;
BEGIN
   Write(PRN, ESC + '*rB');
   Write(PRN, ESC + 'E')
END;

PROCEDURE SluitEpson;
BEGIN
   Write(PRN, #12);
   Write(PRN, #27 + '@')
END;

PROCEDURE HPAfdruk(Dichtheid:Word);
VAR
   REGELLENGTE: String[2];
   I: Integer;

   PROCEDURE PuntenLijn (Y : Word);
   VAR
      REGEL: String[255];
      BASIS: Word;
      BITNR, BYTENR, DATABYTE: Byte;
      KLEUR: Word;

   BEGIN
      REGEL := INTRO;
      FOR BYTENR := 0 TO BREEDTE DO
      BEGIN
         DATABYTE := 0;
         BASIS := 8 * BYTENR;
         FOR BITNR := 0 TO 7 DO
         BEGIN
            KLEUR := GetPixel(BITNR + BASIS, Y);
            IF (KLEUR <> 0) AND (KLEUR > 9) THEN
            DATABYTE := DATABYTE + 128 SHR BITNR
         END;
         REGEL := REGEL + Chr(DATABYTE)
      END;
```

```
      Write(PRN, REGEL)
   END;

BEGIN
   GetViewSettings(VIEWPORT);
   WITH VIEWPORT DO
   BEGIN
      BREEDTE := (X2 + 1) - X1;
      BREEDTE := (BREEDTE - 7) DIV 8;
      HOOGTE  := Y2 - Y1
   END;
   Str(BREEDTE + 1, REGELLENGTE);
   INTRO := ESC + '*b' + REGELLENGTE + 'W';
   InitHP(Dichtheid);
   FOR I := 0 TO HOOGTE + 1 DO PuntenLijn(I);
   SluitHP
END;

PROCEDURE EPAfdruk(Dichtheid:Word);
VAR
   X, Y, Y_OFS: Integer;
   BITGEGEVENS, BITS: Byte;
   KLEUR: Byte;

BEGIN
   GetViewSettings(VIEWPORT);
   WITH VIEWPORT DO
   BEGIN
      HOOGTE  := Y2 - Y1;
      BREEDTE := X2 + 1 - X1
   END;
   InitEpson(Dichtheid);
   Y := 0;
   WHILE Y < HOOGTE DO
   BEGIN
      Write
      (PRN, INTRO, Chr(Lo(BREEDTE)), Chr(Hi(BREEDTE)));
      FOR X := 0 TO BREEDTE - 1 DO
      BEGIN
         BITGEGEVENS := 0;
         IF Y + 7 <= HOOGTE THEN BITS := 7
         ELSE  BITS := HOOGTE - Y;
         FOR Y_OFS := 0 TO BITS DO
         BEGIN
            KLEUR := GetPixel(X,Y_OFS + Y);
            IF ((KLEUR <> 0) AND (KLEUR > 9)) THEN
```

```
            BITGEGEVENS := BITGEGEVENS + 128 SHR Y_OFS
         END;
         Write(PRN, Chr(BITGEGEVENS))
      END;
      Writeln(PRN);
      Inc(Y,8)
   END;
   SluitEpson
END;

PROCEDURE DrukVenster(Dichtheid:Word);
BEGIN
   CASE PRINTERTYPE OF
      HP   : HPAfdruk(Dichtheid);
      EPSON: EPAfdruk(Dichtheid)
   END
END;

BEGIN
   WITH TextRec(PRN) DO
   BEGIN
      Mode      := FmOutput;
      BufSize   := SizeOf (Buffer);
      BufPtr    := @Buffer;
      BufPos    := 0;
      OpenFunc  := @NulFunctie;
      InOutFunc := @UitvoerNaarPrinter;
      FlushFunc := @UitvoerNaarPrinter;
      CloseFunc := @UitvoerNaarPrinter;
      UserData[1] := 0
   END
END.
```

Regels: Toelichting:

3 Gebruik de units CRT, DOS en GRAPH.

[1]5 Definieer de namen van de printertypen.

6-7 Zet de unit standaard op HP.

8-9Declareer een file van het type Text.

10 De procedure DrukVenster is een globale procedure.

11 Begin van het implementatie-gedeelte.

12-17 Declareer een lokale constante en de benodigde variabelen van de unit.

18-23 Function NulFunctie(Rec:TextRec):Integer.

22 NulFunctie retourneert de waarde 0.

[11]24-52 Function UitvoerNaarPrinter(VAR Rec:TextRec):Integer.

27-29 Declareer lokale variabelen van UitvoerNaarPrinter.

[12]31-43 Stuur de regel byte voor byte naar de printer met interrupt hex 17.

[13]44-51Lees het resultaat uit REG.AH en geef de overeenkomstige foutcode aan UitvoerNaarPrinter.

[10]53-70 Procedure InitHP(Dichtheid:Integer).

58-70Geef een HP-printer de startcommando's.

71-84 Procedure InitEpson(Dichtheid:Integer).

74-84 Geef een Epson-printer de startcommando's.

85-89 Procedure SluitHP.

87-88 Draai het papier door en reset de HP-printer.

90-94 Procedure SluitEpson.

92-93Draai het papier door en reset de Epson-printer.

[6]95-134Procedure HPAfdruk(Dichtheid:Word).

96-98 Declareer de lokale variabelen van HPAfdruk.

99-120 Procedure PuntenLijn(Y:Word):procedure van HPAfdruk.

100-104Declareer de lokale variabelen van de procedure PuntenLijn.

[15]106-118Maak een regel van bytes, die samengesteld wordt uit de beeldpunten van het scherm.

[16]119 Stuur de regel naar de printer.

[7]122 Haal de instellingen van het laatst ingestelde venster op.

[8]123-128Reken de breedte van het venster uit in bytes en de hoogte in beeldlijnen en zet de gevonden waarden in de overeenkomstige variabelen.

[9]129-130Stel de regel samen die vooraf moet gaan aan iedere regel die naar de printer gestuurd wordt.

131 Roep de procedure InitHP aan.

[14]132 Ga een FOR-lus in die doorlopen wordt tot alle beeldlijnen zijn verwerkt. Roep bij iedere doorloop de procedure PuntenLijn aan.

133 Roep de procedure SluitHP aan.

[17]135-170 Procedure EPAfdruk(Dichtheid:Word).

136-139 Declareer lokale variabelen voor EPAfdruk.
141 Haal de gegevens van het laatst ingestelde venster op.
143-146 Reken hoogte en breedte van het scherm uit in bits.
147 Roep de procedure InitEpson aan.
149-168 Stel uit de bits op acht beeldlijnen zoveel bytes samen tot de waarde in BREEDTE bereikt is. Stuur iedere byte naar de printer. Als het einde van de regel bereikt is, wordt met Writeln naar de volgende regel gesprongen. Onderzoek de volgende acht beeldlijnen. Dit wordt herhaald tot alle beeldlijnen verwerkt zijn.
169 Roep de procedure SluitEpson aan
171-177 Procedure DrukVenster(Dichtheid:Word).
[5]172-177 Als de printer van het type HP is, roep dan HPAfdruk aan. Is de printer van het type Epson, ga dan naar EPAfdruk.
[2]178-191 Initialiseringsdeel van de unit.
[3]179 Maak met de typecasting TextRec(PRN) het gegevensblok van PRN toegankelijk.
[4]181-189 Geef de velden van het gegevensblok zodanige waarden, dat de functies van GRPRINT.PAS gebruikt worden voor in- en uitvoerbewerkingen.

Toelichting:

[1]In GRPRINT.PAS wordt het type PrinterTypen gedeclareerd. Dit type kan slechts de waarden HP en EPSON bevatten. De getypeerde constante PRINTERTYPE krijgt standaard de waarde HP. Omdat het hier een getypeerde constante betreft, kan deze waarde in een programma veranderen worden. Je zou ook een printerkeuze kunnen maken door de gewenste printer bij de start van een programma uit een bestand te lezen. Op deze manier kan de gebruiker van het programma zelf het printertype instellen. Als de unit ook geschikt gemaakt wordt voor andere typen printers, kun je het aantal mogelijke waarden natuurlijk uitbreiden.

Als globale variabele wordt de file PRN gedeclareerd. Deze file is van het type Text.

[2]Bij het starten van een programma dat in de USES-regel de unit GRPRINT.PAS vermeldt, wordt eerst de initialisering van de unit afgewerkt. Het initialiseringsdeel kan worden vergeleken met het hoofdprogramma van een Turbo Pascal-programma. Het staat onderin de unit en start met een BEGIN en eindigt met een END, gevolgd door een punt.

[3]Met de typecasting TextRec(PRN) wordt het gegevensblok van de tekstfile PRN toegankelijk. TextRec is een in de unit DOS gedeclareerd record.

[4]Het veld Mode wordt op FmOutput gezet. Dit is wat er gebeurt als er een Rewrite-commando gegeven zou worden. Turbo Pascal heeft een aantal constanten om de mode in te stellen:

Naam:	Waarde:
FmClosed	hex D7B0
FmInput	hex D7B1
FmOutput	hex D7B2
FmInOut	hex D7B3

De namen van de constanten geven duidelijk hun betekenis aan. De mode FmInOut wordt gebruikt bij willekeurig toegankelijke bestanden (random access).

De volgende drie velden van TextRec hebben betrekking op de vraag welke buffer gebruikt zal worden. In dit geval wordt gekozen voor de standaardbuffer. We zouden deze velden ook kunnen laten wijzen naar een zelfgedefinieerde, grotere buffer.

Het adres dat in het veld OpenFunc geplaatst wordt, is het adres van de functie NulFunctie in de unit GRPRINT.PAS. Het enige wat deze functie doet, is een 0 retourneren. Dit heeft tot gevolg dat IOResult, de Turbo Pascal-variabele die uitgelezen kan worden na een in- of uitvoerbewerking, de waarde 0 krijgt.

De drie velden die hierna volgen krijgen allemaal het adres van de procedure UitvoerNaarPrinter. Het gevolg hiervan is, dat in- en uitvoerbewerkingen nu naar de eigen procedure UitvoerNaarPrinter gestuurd worden. Dit is nodig, omdat een textfile een EOF-markering (ASCII 26) herkent als teken voor het einde van het bestand. Bij het naar de printer sturen van grafische bestanden geldt dit natuurlijk niet. De procedure UitvoerNaarPrinter stuurt ieder teken gewoon door. Ten slotte wordt in UserData in het eerste element de waarde 0 gezet. Deze waarde staat voor de printerpoort waar de tekens naartoe gestuurd moeten worden. DOS telt vanaf 0. Als er meer printers met je computer verbonden zijn, hoeft alleen het nummer in UserData veranderd te worden om de tekst naar een andere poort en dus naar een andere printer te sturen.

[5]De procedure DrukVenster is de procedure die de unit GRPRINT.PAS toegankelijk maakt. Deze procedure is gedeclareerd als globale procedure. Alle andere functies en procedures van de unit zijn lokaal. Met andere woorden: zij zijn slechts vanuit de unit bereikbaar. De procedure DrukVenster doet eigenlijk maar één ding: kijken of de printer van het type HP of Epson is. Afhankelijk van het printertype wordt de bijbehorende procedure aangeroepen. We gaan er nu van uit dat het een HP-printer betreft. We springen dus naar de procedure HPAfdruk.

[6]De procedure HPAfdruk moet onderzoeken welke beeldpunten een afwijkende kleur ten opzichte van de achtergrondkleur hebben. Zo'n beeldpunt moet dan afgedrukt worden. In deze procedure wordt ieder beeldpunt van het laatst gedefinieerde venster bekeken en wordt beoordeeld of het beeldpunt wel of niet afgedrukt moet worden.

[7]Eerst worden de coördinaten van het laatst ingestelde venster opgevraagd. Hier wordt de GRAPH-procedure GetViewSettings gebruikt. Deze procedure krijgt als parameter de variabele VIEWPORT mee. Deze variabele is als lokale variabele van het type ViewPortType gedeclareerd.

[8]De coördinaten die in VIEWPORT staan, worden gebruikt om de hoogte en breedte van het af te tasten venster uit te rekenen. Deze waarden worden in de variabelen HOOGTE en BREEDTE gezet. De breedte van het scherm wordt uitgedrukt in bytes. Hiervoor wordt het aantal horizontale beeldpunten gedeeld door acht. De hoogte wordt uitgedrukt in het aantal verticale beeldpunten.

[9]Iedere grafische schrijfopdracht naar de printer moet voorafgegaan worden door een voor de printer begrijpelijk commando. In de variabele INTRO wordt deze printeropdracht samengevoegd. De correcte printeropdrachten voor jouw printer staan in de handleiding die je bij je printer hebt gekregen.

[10]Voordat de printer een printopdracht kan uitvoeren, moet hij eerst een aantal commando's krijgen voor de juiste instelling. Onder andere wordt hier het aantal afdrukpunten (dots) per inch ingesteld. Een gangbare instelling is 100 dots per inch. Bij een hoger aantal wordt het afgedrukte beeld kleiner. Bij een lager aantal zal het plaatje groter worden.

Je printerhandleiding vermeldt hoe de commando's gegeven moeten worden. Een commando voor de printer begint meestal met het teken 27 uit de ASCII-tabel. Dit teken is hier opgenomen onder de constante ESC.

[11]De Write-commando's gaan automatisch naar de functie UitvoerNaarPrinter, omdat in het veld InOutFunc van het TextRec dat bij de file hoort het adres van de functie UitvoerNaarPrinter staat. Deze functie stuurt een Write- of Writeln-commando door naar de printer. UitvoerNaarPrinter ontvangt als VAR-parameter het gegevensblok dat bij de file hoort. Gegevens uit dit record worden gebruikt om de bytes naar de juiste printer te sturen.

Het veld BufPos wijst naar de plek na de laatste byte in de buffer die geprint moet worden. Let wel op: functies en procedures die op een dergelijke manier gekoppeld worden, moeten

met de optie {$F+} gecompileerd worden.

[12]De WHILE-lus die nu ingegaan wordt, houden we aan tot WIJZER, die bij iedere doorloop verhoogd wordt, gelijk is aan BufPos, tenzij er een fout optreedt. Als WIJZER de waarde van BufPos bereikt heeft, is de inhoud van de buffer naar de printer gestuurd. Interrupt hex 17 stuurt een byte naar de printer. Het register AH moet hierbij op 0 staan en in het register AL staat het nummer van het te verzenden teken.

In het eerste element van UserData staat het nummer van de poort waar de printer op aangesloten is. Als de regel verwerkt is, wordt BufPos op 0 gezet.

[13]Na uitvoering van de interrupt staat in register AH het resultaat van de bewerking.
Als de bewerking foutloos verlopen is, staat bit drie op 1. Als de bewerking "AH AND 16" de uitkomst 16 heeft, dan is de operatie dus foutloos verlopen. Als de uitkomst van de bewerking 0 is, dan is er een fout gemaakt. Als de uitkomst wel 16 is, dan retourneert UitvoerNaarPrinter een 0. Als bit 5 op 1 staat, dan is het papier op en retourneert UitvoerNaarPrinter DOS-fout 159. Anders wordt de algemene DOS-fout 160 geretourneerd.

[14]We keren terug in de procedure HPAfdruk en gaan een FOR-lus in die voor iedere beeldlijn de lokale procedure PuntenLijn aanroept.

[15]De breedte van het venster is uitgedrukt in bytes. De FOR-lus werkt deze bytes door. Voor iedere byte moet nu gekeken worden of de bits in de byte aan- of uitgezet moeten worden. Het begin van de byte wordt in de lokale variabele BASIS gezet.

Nu gaan we een geneste FOR-lus in, die met behulp van GetPixel de acht beeldpunten vanaf het adres in BASIS bekijkt. GetPixel retourneert het elementnummer van het geldige palet. Als dit ongelijk aan 0 is, hebben we te maken met een beeldpunt dat een andere kleur heeft dan de achtergrondkleur.

Om niet een aantal zwart ingekleurde vlakken te krijgen, hebben we hier tevens aangeven dat de kleur groter dan 9 moet zijn. Als je vervolgens de lijnen trekt met kleuren uit het palet groter dan 9, worden alleen de lijnen zichtbaar.

Het opvullen van de vlakken met verschillende patronen kan met SetFillStyle. Als het onderzochte beeldpunt aangezet moet worden, wordt het overeenkomstige bit in de variabele DATABYTE op 1 gezet.

[16]Nadat de byte geformeerd is, wordt hij toegevoegd aan de variabele REGEL. Als de hele breedte is verwerkt, wordt de gemaakte regel naar de file PRN geschreven en wordt automatisch de procedure UitvoerNaarPrinter aangeroepen. Nadat alle beeldlijnen verwerkt zijn, kan de procedure SluitHP aangeroepen worden. Deze procedure draait het papier in de printer door en voert een reset uit.

[17]De printopdracht is uitgevoerd en de unit kan verlaten worden. Wij zijn echter nog niet klaar. We moeten nog kijken wat er gebeurt als het niet een HP-printer betreft, maar een Epson. In dat geval wordt de procedure EPAfdruk aangeroepen. Deze procedure werkt eigenlijk op dezelfde manier als de procedure HPAfdruk. Omdat de printercommando's verschillend zijn, worden hier echter de procedures InitEpson en SluitEpson aangeroepen. De parameter Dichtheid wordt hier gebruikt om de grafische kwaliteit in te stellen. De printeropdracht wordt verwerkt in de INTRO.

Printeropdrachten kunnen bekeken worden in de handleiding van de printer. De manier waarop een byte voor de printer samengesteld wordt, verschilt van de manier van HP. Bij Epson-printers worden in horizontale richting, beeldpunt voor beeldpunt, telkens acht beeldlijnen onderzocht. Deze bytes worden naar de printer gestuurd. Aan het begin van iedere regel worden de grafische gegevens naar de printer gestuurd. Naast de INTRO met de grafische gegevens, krijgt de printer de lengte van de regel door. Omdat deze lengte groter dan 255 kan zijn, hebben we hier een type Word voor nodig. Dit type bevat twee bytes. De Turbo Pascal-functies Hi en Lo scheiden de twee bytes van elkaar. De gevonden uitkomsten worden gebruikt om de overeenstemmende tekens uit de ASCII-tabel naar de printer te sturen.

17.13 Drivers en fonts meecompileren

Tot nu toe stonden de drivers en fonts in losse bestanden. Drivers hebben de extensie .BGI en fonts .CHR. Deze bestanden bevinden zich op schijf, en als we ze niet zouden meecompileren, dan moeten ze bij een gecompileerd programma los meegeleverd worden. Deze methode is traag, omdat het programma telkens de gegevens van schijf moet lezen. Borland geeft ons de mogelijkheid deze bestanden in het programma op te nemen.

Bij het pakket Turbo Pascal levert Borland het programma MAKE.EXE mee. Dit programma maakt .EXE of .TPU bestanden aan. Daarnaast levert Borland in de directory "examples\bgi" het bestand BGILINK.MAK mee. In dit bestand staan de gegevens die MAKE nodig heeft om de units BGIDRIV.TPU en BGIFONT.TPU te maken. Dit zijn units die in een programma in de USES-regel kunnen worden opgenomen. Als deze units gebruikt worden, hoeven de losse bestanden niet meer meegeleverd te worden. Je programma zal bovendien sneller worden, omdat zowel de driver als de fonts zich in het geheugen van de computer bevinden en niet telkens van de schijf gelezen hoeven te worden.

Het maken van deze units is een fluitje van een cent. Ga naar DOS en maak de directory waar de BGI- en de CHR-bestanden en BGILINK.MAK zich bevinden, tot de geldige directory. Zorg ervoor dat het programma MAKE.EXE via het pad bereikbaar is.

Tik vervolgens achter de DOS-prompt in:

make -fbgilink.mak

Nadat MAKE is uitgevoerd, zullen de units op de schijf staan.

295

295

295

375

375

375