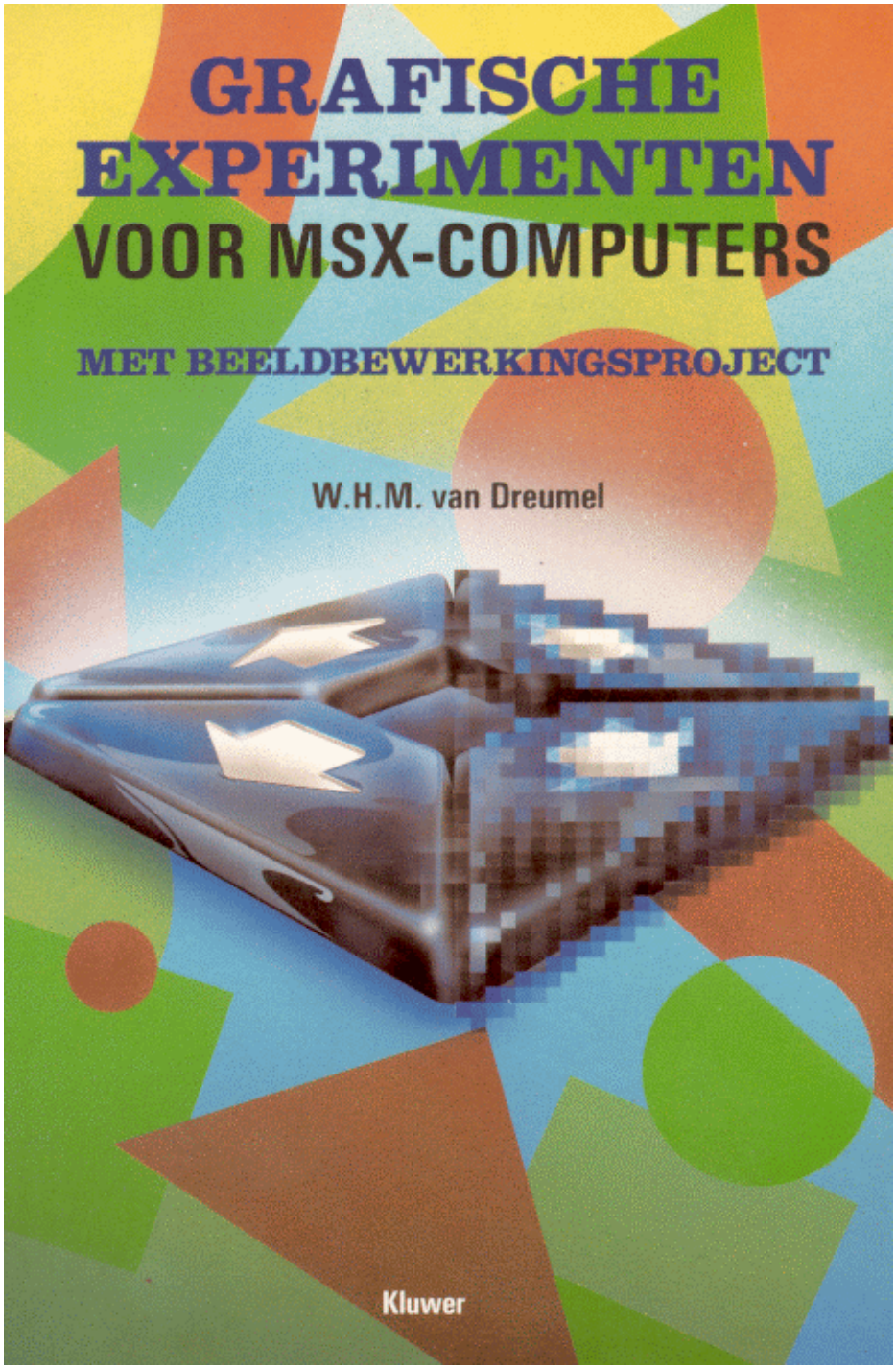

GRAFISCHE
EXPERIMENTEN
VOOR MSX-COMPUTERS
MET BEELDBEWERKINGSPROJECT
W.H.M. van Dreumel
Kluwer

Het maken van grafische afbeeldingen is één van de leukste dingen die men op een computer kan doen. De vele MSX-computerbezitters kunnen aan dit boek dan ook een hoop plezier beleven.
'Grafische experimenten voor MSX-computers' is uit twee delen opgebouwd.
Het eerste deel geeft een reeks korte BASIC-programma's die de grafische mogelijkheden van de MSX-computer op speelse wijze illustreren. Verder biedt het een eenvoudig tekenprogramma en een sprite-ontwerpprogramma. De auteur heeft de programma's aan elkaar geregen zodat de uitvoering ervan een imposante MSX-show teweegbrengt.
Het tweede deel beschrijft hoe we een beeldbewerkingssysteem kunnen bouwen. De hardware van dit systeem bestaat uit een beeld-digitizer, de software bestaat uit beeldverwerkingsprogramma's. Met behulp hiervan kunnen foto's en tekeningen – in digitale code – in de computer worden geladen. We krijgen de foto of tekening opgebouwd uit blokjes op het beeldscherm te zien. Deze 'pixelbeelden' kunnen we spiegelen, kleuren, bijwerken, verdubbelen enz. Zo ontstaan er prachtige plaatjes die we bijvoorbeeld als achtergrond in een computerspel kunnen gebruiken.

ISBN 90 201 1967 2

W.H.M. van Dreumel

Grafische experimenten voor MSX-computers

met beeldbewerkingsproject

Deel 1
Grafische toepassingen

Kluwer Technische Boeken B.V.
Deventer- Antwerpen

Inhoud

Deel 1 Grafische toepassingen

Deel 1

Grafische toepassingen

1 Enkele opmerkingen vooraf

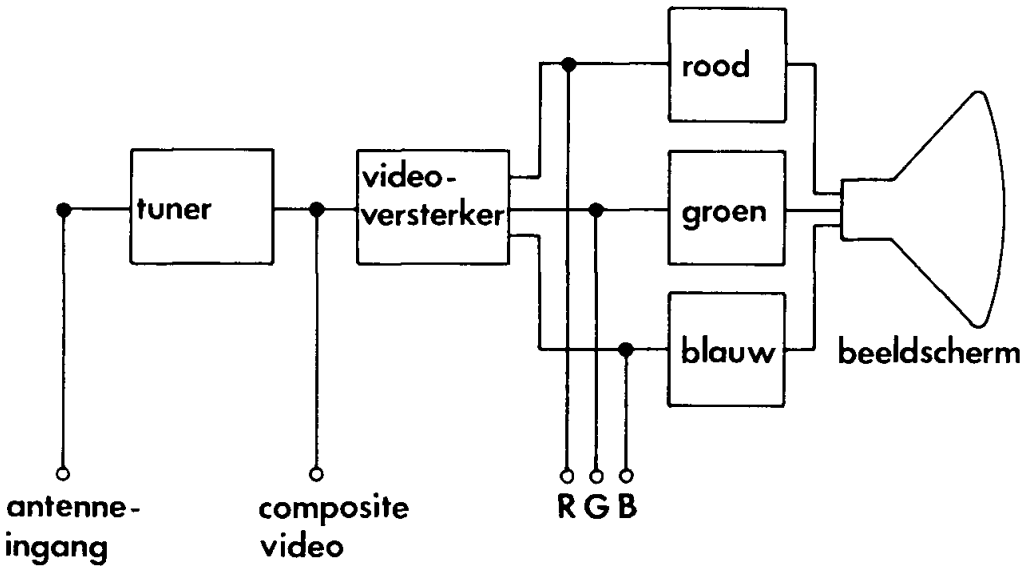
Om te beginnen enkele opmerkingen. Dit boek is geen programmeercursus. Toch worden er heel wat lesjes in programmeren gegeven. Het lezen van dit boek kan dan ook het beste achter de computer gebeuren. Door de programma's direct in te typen en het effect op het scherm te bekijken, zal het grafisch gebruik van de MSX-computer voor u, na het doorwerken van dit deel van het boek, geen geheimen meer hebben. Om het gebruik van de meer ingewikkelde opdrachten eenvoudiger te maken, is een deel ervan apart in hoofdstuk 17 opgenomen. Ze zijn hierbij naar functie gerangschikt. Dus niet alfabetisch zoals in de meeste handleidingen en leerboeken het geval is. Het opzoeken wordt hierdoor vereenvoudigd.

Listings

De manier waarop de programmalistings weergegeven zijn, is enigszins afwijkend. Spaties die normaal gesproken mogen worden weggelaten, zijn in dit boek wél weergegeven. Dit bevordert de overzichtelijkheid van de programma's aanzienlijk zodat minder gemakkelijk typefouten zullen worden gemaakt. Het is niet belangrijk of de afgebeelde spaties wel of niet worden overgenomen. Uw machine regelt zelf wel wat hij ervan gebruikt. Het programma op het scherm kan er dus iets anders uitzien dan de programmalistings in dit boek.

Beeldscherm

Gebruik bij voorkeur een kleurenmonitor of een kleuren-TV met een aparte



Afbeelding 1-1 De aansluitmogelijkheden van een kleurenmonitor.

video-ingang. Op moderne TV's is soms een RGB-plug aanwezig. De MSX-computer heeft hiervoor echter geen standaardaansluiting. Afbeelding 1-1 laat het verschil zien tussen de diverse aansluitmogelijkheden. Hoe minder TV-onderdelen het computersignaal hoeft te passeren, hoe beter de beeldkwaliteit zal zijn.

Cassetterecorder

Het opslaan van programma's op cassette gaat bij de MSX-computer bijzonder handig, doordat de recordermotor vanuit de computer kan worden gestuurd. Er is niets op tegen een gewone audiocassetterecorder voor de opslag van programma's te gebruiken. Voordat de MSX-computer gegevens naar de recorder stuurt, geeft hij een inregelsignaal van ongeveer vijf seconden. Gedurende deze tijd kan de automatische opnamesterkteregeling, die in de meeste audiorecorders aanwezig is, zichzelf instellen. De weergavesterkte moet met de hand worden ingesteld. Enig proberen zal in de meeste gevallen een bevredigende stand van de volumeregelaar opleveren. De volumeregeling dient vrij ver opgedraaid te zijn.
Ideaal is het gebruik van een data-recorder. In wezen is er geen verschil met een gewone audiorecorder. De opname- en weergavesterkte zijn in de stand DATA vast ingesteld. In veel gevallen kan de fase van het signaal worden omgedraaid. Enig proberen geeft ook hier snel resultaat. Bij het uitwisselen van bandjes kan het gebeuren dat programma's niet kunnen worden geladen. In bijna alle gevallen kan dit euvel worden verholpen door de stand van de opname/weergavekop te corrigeren. Vrijwel alle draagbare cassetterecorders hebben een klein gat boven de kop dat toegang geeft tot een afregelschroefje. Soms is dit gat pas te vinden als de afsluitklep open staat. Met een kleine schroevendraaier kan het stelschroefje iets worden verdraaid. Dit afregelen kan het best gebeuren terwijl er een normaal voorbespeeld audiobandje draait dat bij voorkeur afkomstig is van een bekend platenlabel. Bij zo'n bandje mogen we ervan uitgaan dat de opname met een goed ingestelde kop is gemaakt. De schroef wordt zodanig afgesteld dat er zoveel mogelijk hoge tonen te horen zijn.

2 Pareltjes

Een parel is een kostbaar kleinood dat de moeite waard is om te bekijken. Een reeks parels echter, geregen tot een snoer, laat de schoonheid van dit natuurproduct extra tot uiting komen.
Kleine programma's vertonen vaak een treffende gelijkenis met parels. Ze zijn niet zelden sterk beeldscherm-georiënteerd. Het visuele aspect is daardoor erg belangrijk. Een verrassend effect in kleur spreekt sterk tot de verbeelding. Een reeks kleinere programma's kan, samengevoegd tot een geheel, het idee van een parelsnoer weerspiegelen. Het volgende voorbeeld illustreert het parel-idee. Een miniatuurprogramma dat op eenvoudige wijze een mooi plaatje maakt.
Het scherm wordt met SCREEN 2 geschikt gemaakt voor grafisch werken.

Kleuren worden in MSX-BASIC aangegeven met een nummer.

0 Transparant (de kleur van de achtergrond)
1 Zwart
2 Groen
3 Lichtgroen
4 Donkerblauw (MSX meldt zich in deze kleur)
5 Lichtblauw
6 Donkerrood
7 Cyaan (heel licht paars-blauw)
8 Rood
9 Lichtrood
10 Donkergeel
11 Lichtgeel
12 Donkergroen
13 Magenta (violet)
14 Grijs (iets onderdrukt wit)
15 Wit (stralend wit)

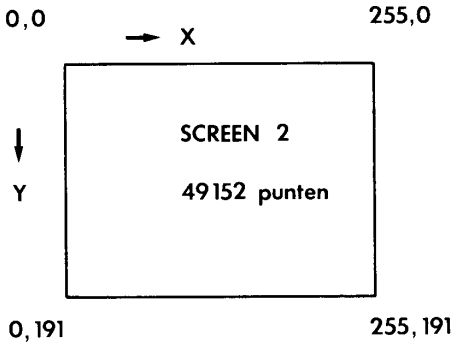
Het lijkt heel wat, maar als we de lijst nauwkeurig bekijken, moeten we constateren dat het eigenlijk tegenvalt:

```
10 REM parel
20 SCREEN 2:COLOR 15,4,5:CLS
30 CIRCLE(128,96),90,14,, ,1.35
40 PAINT(128,96),14
50 GOTO 50
```

Zwart
Wit in twee helderheidsgraden.
Geel in twee helderheidsgraden die erg dicht bij elkaar liggen.
Blauw in twee helderheidsgraden.
Rood in drie helderheidsgraden.

En dan de mengkleuren:
Groen in drie helderheidsgraden.
Cyaan
Violet

In regel 20 van het programma is de inktkleur dus wit (14), de achtergrond donkerblauw (4) en de schermrand lichtblauw (5).Met CLS wordt het scherm gewist en de kleurenopdracht uitgevoerd.Als de opdracht SCREEN 2 wordt gegeven, is het MSX-beeld opgebouwd uit bijna 50.000 beeldpunten. Er zijn 192 horizontale rijen van 256 punten (zie afbeelding 2-1).



Afbeelding 2-1 De opbouw van scherm 2.

De coördinaten (128,96) geven het midden van het schermveld aan. Vanuit dit midden trekt regel 30 een cirkel met een straal van 90 beeldpunten en een grijze lijn (kleur 14). De begin- en de eindhoek van de cirkel worden niet opgegeven („„) zodat er een volledig gesloten cirkel ontstaat. De CIRCLE-opdracht met al zijn varianten krijgt in een later stadium onze uitgebreide aandacht. Het getal 1.35 geeft de rondheid aan. Hoewel de meeste handleidingen aangeven dat er een zuiver ronde cirkel ontstaat als dit getal wordt weggelaten of als het getal 1 is, geeft de door de auteur gebruikte MSX-computer een zuiver ronde cirkel bij het getal 1.35. Mogelijk is dit niet voor alle machines gelijk. Regel 40 vult het cirkelvlak grijs in. Bij het gebruik van PAINT is het noodzakelijk dat de begrenzing dezelfde kleur heeft als de verfkleur. Is dit niet het geval dan loopt de verfkleur over.

Regel 50 zorgt ervoor dat het beeld netjes blijft staan als het programma afgelopen is. Laat hem maar eens weg! De parel krijgt al vorm.
Toch kan een contourlijntje hem nog aardig oppoetsen. En tenslotte maken we er een gaatje in:

```
42 CIRCLE(128,96),70,15,5,5.5,1.35
44 CIRCLE(128,96),70,15,5.6,6,1.35
46 CIRCLE(128,96),50,15,5,5.5,1.35
48 CIRCLE(128)96),50,15,5.6,6,1.35
49 CIRCLE(110,80),8,1,,,1.35
```

Geef de RUN-opdracht (F5) en ziedaar... ons eerste pareltje.Als u een hekel heeft aan de nogal slordige regelnummering die nu is ontstaan, dan typt u gewoon REN 10 in. Als het goed is, verschijnt de list zó op het scherm:

```
10 REM parel
20 SCREEN 2:COLOR 15,4,5:CLS
30 CIRCLE(128,96),90,14,,,1.35
40 PAINT(128,96),14
50 CIRCLEC128,96),70,15,5,5.5,1.35
60 CIRCLEC128,96),70,15,5.6,6,1.35
70 CIRCLEC128,96),50,15,5,5.5,1.35
80 CIRCLEC128,96),50,15,5.6,6,1.35
90 CIRCLEC110,80),8,1,,,1.35
100 GOTO 100
```

Toch best nog ingewikkeld met al die cijfertjes. Vooral het vastleggen van de begin- en eindhoek van een cirkelboog zoals dat in de regels 50 tot en met 80 gebeurd is, zal niet voor iedereen erg doorzichtig zijn. In een later stadium komen we hier echter uitgebreid op terug.
Het volgende programma laat alvast zien dat in een CIRCLE-opdracht niet altijd vaste punten behoeven te worden opgenomen. Afhankelijk van de waarde van I worden steeds nieuwe coördinaten uitgerekend.
Het programma is symbolisch voor het idee achter dit boek. Vele kleintjes kunnen tot een aantrekkelijk geheel worden samengevoegd.

```
10 REM pareltjes
20 SCREEN 2:COLOR 15,4,5:CLS
30 FOR I=1 TO 8
40 CIRCLE(128+15*I,100-I*I),8,,,,1.35
50 PAINT(126+15*I,100-I*I)
60 CIRCLE(128-15*1,100-I*I),8,,, ,1.35
70 PAINT(126-15*I,100-I*I)
80 NEXT I
90 CIRCLE(128,100),8,, , ,1.35
100 PAINT(126,100)
110 GOTO 110
```

De regels 40 en 50 worden herhaald in de regels 60 en 70. Eerst worden alle parels in de rechter beeldhelft getekend waarna de tekenprocedure wordt herhaald voor de linker beeldhelft. Dit komt tot uiting in de + of de - in de X-coördinaat.
Een parelsnoer... of toch niet. Bij nadere bestudering van het scherm blijkt dat de rijgdraad nog ontbreekt. Dit belangrijke onderdeel van het sieraad komt in het volgende hoofdstuk aan de orde.

3 De rijgdraad

Uit de handleiding van de computer blijkt dat er verschillende manieren bestaan om een programma op een cassettebandje te zetten. Bij elke methode van wegschrijven hoort een overeenkomstige manier om het programma van de band weer in het computergeheugen te laden. Er zijn drie van die combinaties met elk een eigen functie.

```
SAVE LOAD
CSAVE CLOAD
BSAVE BLOAD
```

SAVE schrijft een BASIC-programma naar de band, precies zoals het op het scherm wordt weergegeven. Alle tekens worden netjes overgezet in ASCII-co de. Deze code is internationaal vastgelegd.
LOAD haalt het programma terug in het geheugen, ook weer in ASCII-code. Bij het gebruik van SAVE en LOAD wordt het programma als een stuk tekst behandeld. Omdat BASIC-opdrachten altijd hetzelfde zijn, is het niet moeilijk een sterk verkorte code te verzinnen voor alle bestaande BASIC-opdrachten. CSAVE werkt met zo'n sterk verkorte code en is voor het opslaan van BASIC-programma's dan ook het meest geschikt. Het BASIC-programma wordt door het gebruik van codes een flink stuk ingekort. Door te laden met CLOAD worden de ingekorte codes weer op de normale, voor ons leesbare wijze weergegeven.
BSAVE is uitsluitend bedoeld om programma's geschreven in machinetaal op te slaan. Deze methode is dan ook voorbehouden aan de specialist. In dit boek zult u daarom geen enkele machine-instructie tegenkomen. BLOAD laadt een machinetaalprogramma weer in het geheugen. Bij SAVE- en LOAD-opdrachten hoort meestal een toevoeging:

```
SAVE"CAS:tekst" LOAD"CAS:tekst"

BSAVE"CAS:ZAXXON"     BLOAD"CAS:ZAXXON"
```

De toevoeging CAS: geeft aan dat er met de cassette wordt gecommuniceerd. In de CSAVE- en CLOAD-opdracht wordt dat al door de C aangegeven. Uitsluitend de BLOAD-opdracht heeft de mogelijkheid er nog een extra opdracht aan toe te voegen. Althans volgens het handboek.

BLOAD"CAS:ZAXXON",r laadt het machinetaalprogramma ZAXXON van de cassette en start dit automatisch! De 'r' staat dan ook voor RUN. Alleen machinetaalprogramma's kunnen zelfstartend gemaakt worden.

Pareltjes zijn korte programma's die elkaar oproepen. Om pareltjes aan elkaar te kunnen rijgen, moet er iets verzonnen worden om ook gewone BASIC-programma's zelfstartend te maken. We proberen dus wat en schrijven een kort programma:

```
10 CLS:PRINT"Autostart test"
```

We saven het programma op de voorgeschreven manier met CSAVE"test". Let hierbij op het volgende: "Test" is voor de MSX-computer iets anders dan "test" of "TEST". In programmanamen wordt er onderscheid gemaakt tussen hoofdletters en kleine letters. Het is een goede gewoonte om voor programmanamen altijd kleine letters te gebruiken.
We doen net of we de handleiding niet gelezen hebben en proberen tegen beter weten in de autostarttoevoeging 'r'. Spoel het bandje terug en probeer:

```
CLOAD"test",r
```

... jammer, dat werkt dus niet. Moeten alle zelfstartende programma's dan toch in machinetaal worden ontwikkeld? Alleen de gedachte al is zó onaantrekkelijk dat we eerst nog iets anders proberen.

```
10 CLS:PRINT"Autostart test"
```

We schrijven het BASIC-programma weg als een normale tekst. Dus in de onverkorte ASCII-vorm met behulp van:

```
SAVE"test"
```

Als het goed is, gaat de recorder lopen. Bij langere programma's is merkbaar dat het wegschrijven meer tijd kost dan wanneer met CSAVE zou worden weggeschreven. In het kader van onze bedoeling (pareltjes aan een snoertje) levert dat echter geen bezwaar op. Laden proberen we met de bijbehorende opdracht:

`LOAD"test", r` (dus met de autostart-toevoeging)

Dat werkt dus. De tape gaat lopen. Het programma wordt ingelezen. Het scherm wordt gewist en daar verschijnt de tekst 'Autostart test'. Het begin is er. Hoewel dit al aardig is, zijn we er nog niet helemaal. Het is immers onze bedoeling om programma's aan elkaar te rijgen. Een lopend programma moet zelfstandig een volgend programma van de band af laden. Dat programma moet zichzelf starten en daarna op zijn beurt weer het volgende programma oproepen enz. Om te onderzoeken of dat mogelijk is, maken we twee korte programma's.

Eén hebben we al bijna af:

```
10 REM test1
20 CLS:PRINT"Autostart test"
30 LOAD"test2",R
```

Dit programma wordt op tape gezet met:

```
SAVE"test1
```

Het tweede programma luidt:

```
10 REM test2
20 SCREEN 2:COLOR 15,4,4:CLS
30 CIRCLE (130,100),80
40 GOTO 40
```

Ook dit programma wordt op de band gezet, in dit geval met:

SAVE"test2"

Nu komt de klap op de vuurpijl. De band wordt teruggespoeld tot voor het eerste programma. We typen de opdracht:

```
LOAD"test1",r
```

De recorder gaat lopen. Het scherm wordt gewist. De tekst 'Autostart test' verschijnt in beeld. De recorder start nogmaals. Het scherm wordt weer gewist en er verschijnt een witte cirkel in een blauw veld. Het kan dus. We kunnen nu elk BASIC-programma geschikt maken om een volgend programma van de band in het geheugen te laden en uit te laten voeren. Het rijgsnoer is klaar... of kan het nog mooier? Toch wel:

`RUN" test2"` voldoet ook als laadopdracht.

Vooropgesteld dat het programma met SAVE is opgeslagen. Een kleine handicap is dat de naam van het volgende programma bekend moet zijn op het ogenblik dat we de RUN-opdracht in de laatste programmaregel opnemen. Natuurlijk kunnen we alle programma's eenzelfde naam geven. Eenvoudiger is de volgende truc:

```
RUN"CAS:"
```

Deze opdracht kijkt naar het volgende programma op de band, ongeacht de naam en maakt het zelf-startend.
We kennen nu drie manieren om een programma zelf zijn opvolger tot actie aan te sporen.

```
LOAD"naam",r
RUN"naam"
RUN"CAS:"
```

In alle gevallen moet het programma met

```
SAVE"naam"
```

op de band zijn gezet.

We gebruiken de eenvoudigste:

```
RUN"CAS:"
```

We gaan aan de slag. Binnen een uur kunt u al een aardige show op het scherm presenteren.

4 Showtime

Een show begint met een entree. Toeters, bellen en glitter. Het onderstaande programma brengt dat op bescheiden wijze tot uiting.

```
10 REM entree
20 SCREEN 3:COLOR 10,4,4:CLS
30 PLAY"T32O5E","T32O4A","T32O6E"
40 OPEN"GRP:" FOR OUTPUT AS #1
50 PSET(80,30),4
60 PRINT#1,"MSX"
70 PSET(65,90),4
80 COLOR 13
90 PRINT#1,"SHOW"
100 PSET(65,160),4
110 COLOR 3
120 PRINTt#1,"TIME"
130 RUN"CAS:"       (130 GOTO 130)
```

Bij het intypen van de programma's uit dit boek, maar ook bij het zelf ontwikkelen van zelfstartende programma's, is het handig om de laatste regel pas toe te voegen als het programma goed werkt. In plaats van het volgende programma op te starten, kunnen we het programma ook laten pauzeren door in de laatste regel een oneindige lus op te nemen. (In de programmalistings staan beide mogelijkheden. Typ dus één van beide in.) Dit om te voorkomen dat de recordermotor per ongeluk wordt gestart als het programma tijdens proefdraaien de laatste regel bereikt. Er staat namelijk nog geen volgend programma op de band. Bij het rijgen van een parelsnoer zijn we altijd met de laatste parel bezig. Het is natuurlijk ook mogelijk de recorder nog niet op PLAY (PLAY/ LOAD bij DAT A-recorders) te zetten.
Als het programma zonder fouten in het geheugen staat, moet het op de cassette worden gezet:

```
SAVE"entree"
```

Het losse pareltje dat we al eerder gemaakt hebben zetten we er direct achter, nu uiteraard met de intussen welbekende laatste regel.

```
10 REM parel
20 SCREEN 2:COLOR 15,4,5:CLS
30 CIRCLE(128,96),90,14J, ,1.35
40 PAINT(128,96),14
50 CIRCLE(128,96),70>15,5,5.5,1.35
60 CIRCLE(128,96),70,15,5.6,6,1.35
70 CIRCLE(128,96),50,15,5,5.5,1.35
80     CIRCLE(128,96),50,15,5.6,6,1.35
90     CIRCLE(110,80),8,1,,,1.35
100 RUN"CAS:" (100  GOTO  100)
```

Vergeet niet bij het wegschrijven naar tape de recorder op RECORD (REC/SAVE) te zetten.
Tussen de programma's wordt geen extra pauze opgenomen. Ze komen direct achter elkaar. Als er een nieuw pareltje klaar is, wordt de opnametoets ingedrukt. De SAVE-opdracht start dan vanzelf de recorder. Het allerlaatste programma aan het parelsnoer zal de recorder starten op zoek naar een opvolger. Als niemand ingrijpt, loopt het bandje gewoon verder. Niets aan de hand dus, want aan het eind slaat de recorder netjes af. Voor we ons wat systematischer met pareltjes gaan bezighouden, voegen we - om de vaart erin te houden - eerst nog een ander, al eerder beschreven miniprogramma toe.
Na de MSX-entree en de losse parel kunnen we toch niet zonder een afbeelding van een compleet snoer. Toen we dit programma eerder behandelden, waren we de rijgtechniek nog niet meester. Nu kunnen we gaatjes boren en aanrijgen.

```
10 REM pareltjes
20 SCREEN  2:COLOR  15,4,5:CLS
30 FOR  1=1  TO  8
40   CIRCLE(128+15*1,100-1*1),8,,, ,1.35
50   PAINT(126+15*I,100-1*1)
60   CIRCLE(128-15*1,100-1*1), 8,,,,1.35
70   PAINT(126-15*I,100-1*1)
80 NEXT  I
90 RUN "CAS:"            (90  GOTO  90)
```

Stoort u zich aan het feit dat het rijgsnoer zelf nog steeds ontbreekt? Wat let u er zelf een te programmeren. Tip: maak een witte cirkel met een grote straal. Leg het middelpunt ergens in het midden boven het scherm. Dit wordt bereikt door een negatieve Y-coördinaat te kiezen.
Zoals reeds eerder gesteld, is dit boek geen cursus in programmeertechnieken. Toch zullen we bij het spelen met de MSX-computer zeer veel aspecten tegenkomen. Om het terugzoeken van onderwerpen in dit boek een beetje gemakkelijk te maken, zullen we enige lijn proberen aan te brengen. De technieken zullen enigszins worden gegroepeerd, zonder de grenzen erg scherp te leggen. Het spelelement blijft voorop staan. Mocht u niet geïnteresseerd zijn in de opbouw van de programma's dan kan de tussen de programma's opgenomen tekst natuurlijk gewoon worden overgeslagen. Het is best leuk om de parels alleen maar in te typen en aan elkaar te rijgen. Miniprogramma's uit tijdschriften kunnen simpel worden toegevoegd.

5 Tekenen op het tekstscherm

Als de computer wordt aangezet, is scherm 0 te zien. Op dit scherm van 24 lijnen met 40 tekenposities kunnen geen grafische opdrachten uitgevoerd worden. Datzelfde geldt voor scherm 1, dat 24 regels heeft met 32 iets grotere tekenposities. Toch wil dat niet zeggen dat op deze tekstschermen geen mooie plaatjes kunnen worden gemaakt. De MSX-computer beschikt immers over een zeer uitgebreide reeks voorgeprogrammeerde figuurtjes. Een aantal is via het toetsenbord op te roepen met behulp van de GRAPH-toets, bij een deel ervan dient ook de SHIFT-toets te worden ingedrukt. De rest kan met de CODE-of SHIFT/CODE-toets worden opgeroepen.

We proberen het gebruik van de ingebouwde grafische tekenset even:

```
10 REM graphics op scherm 0
20 SCREEN 0:COLOR 15,4,4:CLS
30 KEY OFF:WIDTH 40
40 FOR I=1 TO 480
50   PRINT"%&";
60 NEXT I
70 LOCATE 16,11
80 PRINT"@@@@@@@@"
90 RUN"CAS:"                    (90 GOTO 90)
```

Regel 50: [GRAPH][Q] en [GRAPH][q]
Regel 80: acht keer [GRAPH][[]

Dat valt tegen. De figuurtjes overlappen elkaar een stukje. Dat is erg goed te zien bij het smile-gezichtje. Een verklaring hiervoor hoeft niet ver te worden gezocht. Op scherm 0 heeft elk teken een breedte van 6 puntjes. De grafische tekens zijn echter ontworpen met een breedte van 8 puntjes. Scherm 0 is dus echt niet geschikt voor grafisch werk. Het bovenstaande programma is gemakkelijk om te bouwen voor scherm 1.

```
10 REM graphics op scherm 1
20 SCREEN 1:COLOR 15,4,4:CLS
30 KEY OFF:WIDTH 32
40 FOR I=1 TO 384
50   PRINT"%&";
60 NEXT I
70 LOCATE 12,11
80 PRINT"@@@@@@@@"
90 RUN"CAS:"      (90 GOTO 90)
```

We zien nu een prachtig aaneengesloten visgraatmotief. Ook de smile-gezichtjes zijn helemaal afgebeeld.

Scherm 1 leent zich dus best wel voor grafisch werk. Er kunnen zelfs tekenfilmpjes op worden gemaakt. In een paar stappen zullen we een tekenfilmpje opbouwen. We beginnen met de hoofdrolspeler:

```
10 REM eend
20 SCREEN 1:CLS:KEY OFF
30 WIDTH 32
40 PRINT"    □<
50 PRINT"-■%■-
```

Regel 40: vier keer [spatie], [GRAPH][J],[<]
Regel 50: [GRAPH][0],[GRAPH][P],[GRAPH][q],[GRAPH][P],[GRAPH][O]

Eendje, ga je mee? We geven het diertje een ander plaatsje op het scherm :

```
10 REM eend
20 SCREEN 1:CLS:KEY OFF
30 WIDTH 32
40 X=10:Y=10
50 LOCATE X,Y:PRINT"       □<
60 LOCATE X,Y+1:PRINT"-■%■-
```

Bewegen is nu ook niet moeilijk meer. We passen regel 40 aan en voegen twee regels toe:

```
10 REM turbo eend
20 SCREEN 1:CLS:KEY OFF
30 WIDTH 32
32 Y=2
40 FOR X=1 TO 32
50   LOCATE X,Y:PRINT"       □<
60   LOCATE X,Y+1:PRINT"-■%■-
70 NEXT X
```

Dat gaat hard. Heeft u al eens eerder een eend met raketvoortstuwing gezien? Om het spoo rte wissen, wordt er een extra spatie opgenomen a Is eerste teken van de twee PRINT-opdrachten.

```
50   LOCATE X,Y:PRINT"       □<
60   LOCATE X,Y+1:PRINT"-■%■-
```

Dat ziet er al een stuk beter uit. Maar het lijkt nog steeds meer op een kanonskogel dan op een levend diertje. We leren hem vliegen.

```
10 REM turbo eend met vleugels
20 SCREEN 1:CLS:KEY OFF
30 WIDTH 32
32 Y=2
40 FOR X=1 TO 32 STEP 2
50 LOCATE X,Y:PRINT"       ◘<
60 LOCATE X,Y+1:PRINT"-■▒■-
70 LOCATE X+1,Y:PRINT"     ▒ ◘<
80 LOCATE X+1,Y+1:PRINT"-■■■-
90 NEXT X
```

Regel 70: drie keer [spatie],[GRAPH][Q],[spatie],[GRAPH[J],[<]

De regels 50 en 60 zijn met een kleine verandering herhaald in 70 en 80. De verandering zit niet alleen in de verhoging van de waarde X met 1. Het eendje heeft bovendien een vleugeltje gekregen. Ook regel 40 heeft een kleine uitbreiding ondergaan. Hij vliegt, maar ach jee wat een haast! Twee extra regeltjes zorgen ervoor dat we de vliegsnelheid beheersen.

62 FOR Q=l TO 100:NEXT Q
82 FOR Q=l TO 100:NEXT Q

Deze lus wordt in programma's vaak gebruikt om wat tijd te rekken. Het getal geeft aan hoeveel keer de computer in deze lus rond moet rennen voor hij verder mag. Vul er maar eens wat anders voor in. Wat dacht u van een slootkant?

```
33 FOR I=0 TO 10
34   LOCATE I,16:PRINT"*"
35   LOCATE I,17:PRINT"┼"
36   LOCATE I,18:PRINT"┴"
37 NEXT I
38 LOCATE 0,19:PRINT STRING$(32,215)
```

Regel 34: [GRAPH][Z]
Regel 35: [SHIFT][GRAPH][G]
Regel 36: [GRAPH][B]

Nu komt het probleem van scherm 1 aan het licht. Het is zonder kunstgrepen niet mogelijk om meer kleuren af te beelden. Bij gebruik van de opdracht COLOR 10 worden zowel de eend als de slootkant geel. Deze beperking doet ons ertoe besluiten de tekstschermen O en 1 verder links te laten liggen. Echter niet voordat het werk dat we in dit hoofdstuk gedaan hebben als een echt pareltje aan de ketting is geregen. Eerst hernummeren we het programma met RENUM, dan voegen we de kettingaansluiting toe. Als alles goed gegaan is, ziet het er ongeveer zó uit:

```
10  REM turbo eend met vleugels
20  SCREEN 1:COLOR 15,4,4:CLS:KEY OFF
30  WIDTH 32
40  Y=2
50  FOR I=0 TO 10
60    LOCATE I,16:PRINT"*"
70    LOCATE I,17:PRINT"+"
80    LOCATE I,18:PRINT"⊥"
90  NEXT I
100 LOCATE 0,19:PRINT STRING$(32,215)
110 FOR X=1 TO 32 STEP 2
120   LOCATE X,Y:PRINT"      ◘<"
130   LOCATEX,Y+1:PRINT"▀▄▒▄▀ "
140   FOR Q=1 TO 100:NEXT Q
150   LOCATE X+1,Y:PRINT"     ▒ ◘<
160   LOCATE X+1,Y+1:PRINT"▀▄▄▄▀ "
170   FOR Q=1 TO 100:NEXT Q
180 NEXT X
190 RUN"CAS:"           (190 GOTO 190)
```

Het zal duidelijk zijn dat de techniek die toegepast is om het eendje te laten vliegen ook benut kan worden om teksten over het scherm te laten bewegen. Maak zelf eens een bewegende lichtkrant en rijg hem aan de ketting met:

```
RUN"CAS:"
```

6 Scherm 3

Hoewel de term 'lage resolutie' doet vermoeden dat er nogal wat beperkingen aan scherm 3 kleven, pakt de praktijk anders uit. Geringschattende opmerkingen over de slechte kwaliteit van de op dit scherm weer te geven beelden komen voort uit onbekendheid of uit gebrek aan fantasie. In tegenstelling tot de tekstschermen, waarop slechts in één kleur kan worden gewerkt en het hoge-resolutiescherm dat later aan de orde komt, kan bij scherm 3 ieder beeldpunt elke kleur aannemen die de MSX-computer tot zijn beschikking heeft. Goed, de beeldpunten zijn wat groot uitgevallen, maar voor sommige beelden is dat juist prachtig. Het geeft de plaatjes een echt computeraspect. Kijk maar:

```
10  REM msx
20  SCREEN 3,3:COLOR 15,1,1:CLS
30  FOR 1=0 TO 2
40    I$=""
50   FOR S=0 TO 31
60     READ S$
70      I$=I$+CHR$(VAL("&H"+S$))
80    NEXT S
90    SPRITE$(I)=I$
100 NEXT I
110 DATA 00,00,00,00,00,00,00,00,00,00,
         00,00,00,01,01,00
120 DATA 00,00,00,00,39,39,7D,7F,7F,7F,
         FF,FF,EF,EF,07,00
130 DATA 00,00,00,00,C7,CF,CF,EE,EF,EF,
         F7,FO,7F,7F,3F,00
140 DATA 00,00,00,00,FE,FF,FF,03,F1,F8,F9,
         3B,FF,FF,FF,00
150 DATA 00,00,00,00,07,OF,9E,FC,F8,F0,F8,
         FG,DE,8F,07,00
160 DATA 00,00,00,00,80,00,00,00,00,00,00,
         00,00,00,00,00
170 FOR X=l TO 80
180   Y=X*4/5
190   PUT SPRITE 0,(X,Y),8,0
200   PUT SPRITE l,(X+32,Y),8,l
210   PUT SPRITE 2,(X+64,Y),8,2
220 NEXT X
230 CIRCLE(128,85),40,10
240 COLOR,,4
250 OPEN"GRP:" FOR OUTPUT AS #1
260 PSET(7,150),l
270 COLOR 12
280 PRINT#l,"COMPUTER"
290 RUN"CAS:" (290 GOTO 290)
```

Het programma loopt vooruit op een aantal zaken die nog uitvoerig aan de orde zullen komen, zoals het maken van sprites, tekenen van geometrische figuren en teksten op het grafische scherm. Het programma illustreert dat afbeeldingen op het lageresolutiescherm best wel aantrekkelijk kunnen zijn. De ketting bevat nu al een aantal pareltjes. Spoel het bandje maar eens helemaal terug en start het met

```
RUN"CAS:"
```

En dit is nog maar het begin.
Een volgend pareltje laat alle kleuren zien die de MSX-computer kan weergeven. Een klein programma met een bont resultaat.

```
10 REM ruit
20 SCREEN 3:CLS
30 FOR X=0 TO 256 STEP 4
40  FOR Y=0 TO 192 STEP 4
50    C=C+1
60    IF C=16 THEN C=0
70    PSET(X,Y),C
80 NEXT Y,X
90 RUN"CAS="  (90 GOTO 90)
```

Uit dit programma blijkt duidelijk hoe de beeldopbouw op het lageresolutiescherm plaatsvindt. Het scherm kent even veel beeldpunten als het hogeresolutiescherm 2. Namelijk 256 x 192. Echter, als een enkel beeldpunt wordt aangewezen, dan maakt de computer een blokje van 4x4 beeldpunten. In de regels 30 en 40 wordt dan ook horizontaal en verticaal met stapjes van vier beeldpunten gesprongen. Regel 70 doet het eigenlijke werk. Hij wijst het punt aan waar het blokje komt dat moet worden gekleurd. De kleur wordt in regel 50 steeds veranderd. Omdat kleurnummer 16 niet bestaat, laat regel 60 de kleurtelling steeds opnieuw beginnen. Een aardig effect, nietwaar? Een variant wordt in het volgende stukje beschreven.

```
10 REM lapjesdoek
20 SCREEN 3:COLOR ,1,1:CLS
30 FOR X=0 TO 256 STEP 4
40  FOR Y=0 TO 192 STEP 4
50    C=RND(1)*16
60    PSET(X,Y),C
70 NEXT Y,X
80 RUN"CAS:"  (80 GOTO 80)
```

In MSX-BASIC mogen niet-gebruikte variabelen gewoon worden weggelaten. Zo ontbreekt het eerste getal in de COLOR-opdracht. Deze inktkleur wordt immers apart vastgelegd als er in regel 60 een punt wordt afgebeeld. In tegenstelling tot het vorige programma wordt de inktkleur door regel 50 willekeurig bepaald. Maar waarom staat er 16, er zijn toch maar 15 kleuren? RND(1) levert een getal tussen 0 en 1. De 1 komt nooit voor. Het maximale getal voor C kan

daarom ook geen 16 worden, maar hoogstens 15,999999. In PSET wordt het getal afgebroken zodat er netjes 15 overblijft. Vergeet regel 80 niet! Teksten kunnen niet zonder meer met de gebruikelijke PRINT-opdracht op een grafisch scherm worden gezet. Probeer het maar met het volgende programma:

```
10 SCREEN 3:CLS
20 PRINT "Dit is een tekst"
RUN
```

De foutmelding geeft aan dat er iets niet goed zit. De handleiding vertelt dan ook dat er geen tekens worden geaccepteerd op de grafische schermen 2 en 3. Dat lijkt een geweldig grote handicap. In veel figuren willen we immers ook met teksten werken. MSX-BASIC heeft echter een zeer flexibele mogelijkheid om teksten in elke gewenste kleur op elke gewenste plaats op het scherm te zetten. We hoeven hiervoor slechts een kanaal naar het grafische scherm te openen. De MSX-computer verlangt wel dat we zo'n kanaal een eigen nummer geven. Bijvoorbeeld: open kanaal nummer 1 voor uitvoer naar het scherm. Of, in echt BASIC:

```
OPEN   "GRP:"   FOR  OUTPUT  AS  #1
```

Hiermee wordt het bovenstaande mislukte programma aangepast.

```
10 SCREEN 3:CLS
20 OPEN "GRP:" FOR OUTPUT AS #1
30 PRINT #l,"Dit is een tekst"
RUN
```

In de PRINT-opdracht wordt tot uiting gebracht dat de te printen tekst voor kanaal 1 bedoeld is.
Zo, het probleem lijkt opgelost. Hoewel, zijn de letters niet wat groot uitgevallen? Bij SCREEN 3 worden alle tekens vier keer zo groot dan normaal. Grote letters zijn niet gek voor reclameboodschappen. Het logo van een bekend Amerikaans bedrijf bijvoorbeeld.

```
10 REM logo
20 SCREEN 3:COLOR 10,4,4:CLS
30 OPEN "GRP:" FOR OUTPUT AS #1
40 PSET(35,77),4
50  PRINT #,"DuPont"
60  CIRCLE(124,90),110,,,,.5
70 RUN"CAS:" (70  GOTO  70)
```

Let erop hoe met een PSET-opdracht de plaats wordt aangegeven waar de tekst moet komen. Dit gaat met beeldpuntnauwkeurigheid, maar ook hier
weer afgerond naar blokjes van 4 x 4. De kleur in de PSET-opdracht (4) is gelijk gekozen aan de achtergrondkleur. Onzichtbaar dus. De CIRCLE-opdracht komt

verderop uitgebreid aan de orde. Vergeet geen komma's en punten. Spaties daarentegen zijn minder belangrijk.
Van de grote letters kunnen we dankbaar gebruik maken. Een parelketting met reclameboodschappen of mededelingen is op meters afstand goed te lezen. Als de letters tenminste netjes gerangschikt op het scherm staan. Niet zó dus:

```
10  REM letters
20  SCREEN 3:COLOR ,1,1:CLS
30  A$="   MSX":B$="COMPUTERS"
40  OPEN "GRP:" FOR OUTPUT AS #1
50  FOR I=1 TO 100
60    COLOR RND(1)*16
70    PRINT#1,A$
80    PSET(RND(1)*250,RND(1)*180)
90    PRINT#1,B$
100 NEXT I
110 PSET(80,50):COLOR 1:PRINT#1,"■■■■"
120 PSET(80,75):COLOR 1:PRINT#1,"■■■■"
130 PSET(90,70):COLOR 10:PRINT#1,"MSX"
140 RUN"CAS:"            (140 GOTO 140)
```

Regel 110 en 120: vier keer [GRAPH][P]

Weer een pareltje aan de ketting. Tekens kunnen op alle gebruikelijke manieren, door een geopend kanaal, naar het grafische scherm worden gestuurd. Direct met PRINT#1," ... ". Maar ook indirect met A$=" ... " gevolgd door PRINT#1,A$. Ook ASCII-codes worden geaccepteerd, zoals PRINT#1,CHR$(177). Deze laatste manier is handig bij het gebruik van de voorgeprogrammeerde grafische tekens. In een afdruk van een programma is de ASCII-code altijd mogelijk, terwijl de meeste printers niet in staat zullen zijn om de grafische blokjes af te drukken.
Er schuilt echter een addertje onder het gras. De eerste 31 tekens kunnen niet met CHR$(...) worden opgeroepen. Het is zeker de moeite waard om het volgende programma even in te typen. Maak er meteen een pareltje van.

```
10 REM tekens op het scherm met CHR$()
20 SCREEN 3:COLOR 15,4,4:CLS
30 OPEN "GRP:" FOR OUTPUT AS #1
40 FOR I=0 TO 255
50 PSET(O,10)
60 PRINT#1,I;CHR$(I)
70 FOR Q=l TO 200:NEXT Q
80 CLS
90 NEXT I
100 RUN"CAS:"            (100 GOTO 100)
```

Op het scherm verschijnen achtereenvolgens de ASCII-waarden van alle tekens die de MSX-computer kent. Dat zijn er op een paar na 256. Achter het nummer wordt het desbetreffende teken afgebeeld. De eerste 31 tekens wordn niet afgebeeld, dan volgt het teken voor de spatie CHR$(32), die is dus ook niet te zien, en vervolgens paradeert de gehele tekenset voorbij. Een voorstelling op zich, die weer eens laat zien hoe veelzijdig de MSX-computer is. De snelheid is

te regelen door regel 70 aan te passen. Ter afsluiting van de reeks parels die we nu gemaakt hebben op het lagereso-lutiescherm voegen we er nog twee aan toe.

```
10 REM lijnenspel
20 SCREEN 3:COLOR ,1,1:CLS
30 FOR I=1 TO 255
40 COLOR RND(1)*14
50 LINE(0,0)-(RND(1)*256,RND(1)*192)
60 LINE(256,0)-(RND(1)*256,RND(1)*192)
70 LINE(0,192)-(RND(1)*256,RND(1)*192)
80 LINE(256,192)-(RND(1)*256,RND(1)*192)
90 NEXT I
100 RUN"CAS:"               (100 GOTO 100)
```

Het getal 14 in regel 40 zet grijs (14) en wit (15) buitenspel.
Om op alles voorbereid te zijn, kan het volgende pareltje al worden ingetypt.
Het opgeroepen beeld zal ons ongetwijfeld in de juiste sfeer brengen.

```
10 REM kerstboom
20  SCREEN 3:COLOR ,1,1:CLS
30  FOR I=1 TO 40
40   PSET(RND(1)*250,RND(1)*145),RND(1)*16
50  NEXT I
60  FOR 1=1 TO 30
70   PSET(130-1.5*1,20+4*1),12
80   A$="R"+STR$(3*I)
90   DRAW A$
100 NEXT I
110 LINE(126,145)-(134,165),10,BF
120 PSET(110,165),12
130 DRAW"M150,165M140,192M120,192M110,165"
140 RUN"CAS:" (140 GOTO 140)
```

Genoeg over lage resolutie. Mogelijk heeft u inmiddels zelf wat pareltjes gecreëerd. Neem ze op in de groeiende reeks programmaatjes.

7 Scherm 2

Op scherm 2 kunnen alle registers open. Op uitbundige wijze kunnen we hierop stoeien met punten, lijnen, cirkels en vierkanten. Sprites komen op dit scherm het mooist tot hun recht. Spelenderwijs zullen we in dit deel van het boek leren omgaan met de veelzijdige grafische gereedschappen die de MSX-computer ons biedt. Maar ook de beperkingen komen aan de orde. Hier volgt alvast een voorproefje om als parel aan het snoer te rijgen.

```
10 REM lijn
20 SCREEN 2:COLOR 6,1,1:CLS
30 FOR A=1 TO 8
40 CIRCLE(130,100),A
42 A$="0=A;L60CDEFGABAGFECD"
44 B$="L60DEFGABAGFEDCC"
46 C$="L60EFGABAGFEDCD"
50 PLAY A$,B$,C$
60 NEXT A
70 FOR I=1 TO 100
80   PSET(130,100)
90   LINE-(RND(1)*256,RND(1)*192),RND(1)*14
100 NEXT I
110 RUN"CAS:"                (110 GOTO 110)
```

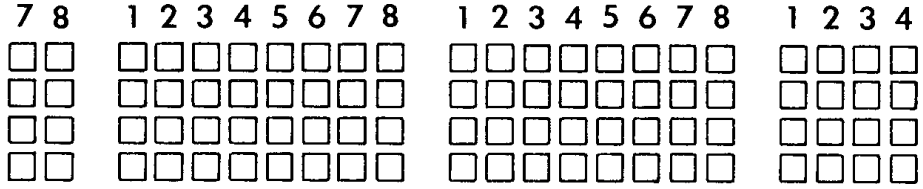
In dit programma komen elementen voor die in de volgende hoofdstukken verder zullen worden uitgewerkt. Maar ook andere grafische opdrachten komen aan de orde. De belangrijkste zijn:

```
PSET (PRESET)
LINE
DRAW
CIRCLE
PAINT
```

Voor we echter overgaan tot een meer uitgebreide toepassing van deze commando's zullen we eerst enkele eigenaardigheden van het hogeresolutie-scherm bekijken. Uiteraard aan de hand van een nieuw pareltje aan de ketting.

```
10 REM probleempje
20 SCREEN 2:COLOR 1,15,15:CLS
30 FOR I=50 TO 200
40   C=C+1:IFC=16 THEN C=0
50   LINE(I,0)-(I,192),C
60   FOR Q=1 TO 300:NEXT Q
70 NEXT I
80 RUN"CAS:"     (80 GOTO 80)
```

Er gebeurt nu iets zeer merkwaardigs. Verticaal worden lijnen getrokken, steeds met een opvolgend kleurnummer. De reeds getrokken lijnen veranderen echter in de kleur van de laatst getrokken lijn. Na acht lijntjes blijft de kleur behouden en begint een volgende baan steeds van kleur te veranderen. In de praktijk zou dit betekenen dat er op de meeste plaatsen van het scherm geen twee verticale lijnen van verschillende kleur naast elkaar kunnen worden afgebeeld. Om dit probleem te kunnen omzeilen, bekijken we in afbeelding 7-1 de schermopbouw van heel dichtbij.



Afbeelding 7-1 Beeldpunten op het monitorscherm.

De beeldpunten zijn per horizontale lijn verdeeld in groepjes van acht. In één zo'n groepje op de horizontale lijnen mogen slechts twee kleuren voorkomen. Stel dat de beeldpunten 1 t/m 7 de achtergrondkleur blauw hebben en dat beeldpunt 8 rood is gekleurd. Als nu één van de blauwe beeldpunten geel wordt gemaakt, dan nemen alle beeldpunten van het desbetreffende groepje de laatst toegevoegde kleur aan: alle punten worden geel. Willen we toch lijnen van verschillende kleur verticaal naast elkaar, dan kan dat alleen maar aan de randen van de groepjes. Bijvoorbeeld rood in beeldpunt 8 en groen in het ernaast liggende beeldpunt 1. De drie kleuren zijn dan netjes verdeeld over twee groepen.
Het volgende programma maakt gebruik van deze handigheid om ruimtelijke staafdiagrammen te maken. Het levert een mooi schermpje op.

```
10 REM foutje weg
20 SCREEN 2:COLOR 15,1,1:CLS
30 LINE(20,0)-(240,192),14,B
40 FOR I=1 TO 4
50   LINE(90-3*I,20+16*I)-(110-
3*I,100+16*I),2*I + 1, BF 60   LINE(216-16*I,20+16*I)-(240-
16*I,100 + 16*I), 2*I + 1, BF
70 NEXT I
80 RUN"CAS:"       (80 GOTO 80)
```

De kolommen die door regel 50 worden getekend, zijn drie beeldpunten ten opzichte van elkaarverschoven. Binnen een groepje van acht horizontale punten kunnen dan ook meer dan twee kleuren voorkomen. De laatst ingegeven kleur overheerst, zoals aan het voorste rode en het op één na achterste blauwe vlak is te zien. Door het geheel iets te verschuiven kan misschien een punt worden gevonden waarbij de schoonheidsfout niet optreedt. Verreweg de handig-

ste methode om het probleem te omzeilen, wordt gedemonstreerd in regel 60. In deze regel is de afstand tussen de kolommen maar liefst 16 beeldpunten (acht zou genoeg zijn).
Eigenlijk is het wonderbaarlijk dat met zo'n eenvoudig programma al aardig complexe 'business-grafieken' kunnen worden gemaakt. Het hier behandelde euvel speelt zich gelukkig alleen maar af langs horizontale groepjes beeldpunten. Verticaal mogen alle beeldpunten verschillend gekleurd zijn. Het volgende stukje BASIC illustreert dat.

```
10 REM groepjes van acht
20  SCREEN 2=COLOR ,l,l:CLS
30  FOR X=50 TO 70
40   FOR Y=50 TO 114
50     IF C=16 THEN C=0
60     PSET(X,Y),C
70     C=C+1
80  NEXT Y,X
90  FOR X=90 TO 110
100   FOR Y=51 TO 114
110    IF C=16 THEN C=0
120    PSET(X,Y),C
130    C=C+1
140 NEXT Y,X
150 RUN"CAS:" (150 GOTO 150)
```

In het linkerdeel van het scherm doet zich het verschijnsel voor dat verticaal alle kleuren wel worden afgebeeld, maar de afzonderlijke horizontale lijntjes verschieten steeds weer van kleur als er een ander kleurtje langs komt. In het rechterdeel liggen de kleuren op gelijke hoogte en ontstaat er een regelmatig regenboogpatroon.
Als u wat programmeerervaring heeft, dan bekruipt u bij het bekijken van het voorgaande programma ongetwijfeld een gevoel van afgrijzen: tweemaal exact dezelfde routine achter elkaar! Om u weer enigszins tot rust te brengen, volgt hieronder een opgepoetste versie, waarbij de te herhalen routine in een subroutine is ondergebracht.

```
10 REM groepjes van acht
20  SCREEN 2:COLOR ,1,1:CLS
30  FOR X=50 TO 70
40   FOR Y=50 TO 114
50     GOSUB 120
60  NEXT Y,X
70  FOR X=90 TO 110
80   FOR Y=51 TO 114
90     GOSUB 120
100 NEXT Y,X
110 RUN"CAS:" (110 GOTO 110)
120 IF C=16 THEN C=0
130 PSET (X,Y),C
140 C=C+1
150 RETURN
```

Tot besluit van dit hoofdstuk over het hogeresolutie scherm volgen nog twee kleine programma's om aan de ketting te rijgen.

```
10 REM lijnenspel
20 SCREEN 2:COLOR ,4,4:CLS
30 FOR I=1 TO 500
40   LINE(0,0)-(RND(1)*256,RND(1)*192),
      RND(1)*16
50 NEXT I
60 RUN"CAS:" (60 GOTO 60)
```

Let vooral eens op de linkerbovenhoek. Het verschijnsel van de acht bij elkaar horende beeldpunten is daar heel goed te zien.

```
10 REM vierkanten
20 SCREEN 2:COLOR ,4,4:CLS
30 FOR I=1 TO 200
40 LINE(0,0)-(RND(1)*256,RND(1)*192),
   RND(1)*16,B
50 NEXT I
60 RUN"CAS:"              (60 GOTO 60)
```

Valt het verschil met het vorige programma op? Inderdaad, afgezien van een kleiner aantal herhalingen heeft de LINE-opdrachtin regel 40 een extra toevoeging. In het volgende hoofdstuk zullen we de LINE-opdracht aan een nader onderzoek onderwerpen.

8 Lijnen en wat daarbij hoort

Het lijkt eenvoudig; met de LINE-opdracht kunnen lijnen worden getrokken. Mooie rechte lijnen, kijk maar:

```
10 REM horizon
20  SCREEN 2:COLOR ,l,l:CLS
30  FOR V=1 TO 6
40   LINE(128-V*8,64)-(120-V*20,191),3
50   LINE(128+V*8,64)-(120+V*20,191),3
60  NEXT V
70  FOR I=1 TO 7
80   READ A
90    LINE(178+I*8,64)-(255,A),3
100   LINE(82-I*8,64)-(0,A),3
110 NEXT I
120 LINE(128,64)-(122,192),3
130 FOR H=l TO 12
140   LINE(0,63+H*H)-(256,63+H*H),3
150 NEXT H
160 DATA 170,142,121,104,90,78,70
170 GOTO 170
```

Dat levert een uitgesproken futuristisch landschap op. En dat met zo'n klein programma. We rijgen dit pareltje aan de ketting, omdat het in de toekomst zeker nog eens van pas zal komen. Het beeld vormt immers een ideaal decor voor een ruimtespel. Vindt u het nog wat statisch? Daar doen we wat aan. In het volgende programma zijn veel regels uit het vorige programma opgenomen. Als u het handig aanpakt, kunt u zich aardig wat typewerk besparen.

```
10 REM horizon 2
20  SCREEN 2:COLOR ,1,1:CLS
30  FOR V=l TO 6
40   LINE(128-V*8,64)-(120-7*20,191), 3
50   LINE(128+V*8,64)-(120+V*20,191),3
60  NEXT V
70  FOR I=1 TO 7:READ A
LINE(l78  +  I*8,64)-(255,A),  3  90     LINE(82-I*8,64)-(0,A),3  100
NEXT I
110 LINE(128,64)-(122,192),3
120 LINE(0,63)-(256,63),3
130 FOR T=1 TO 20
140  FOR H=1 TO 11
150    LINE(0,63+H*H)-(256,63+H*H),1
160  NEXT H
170  FOR I=1 TO 10
```

```
180   PSET(RND(1)*256,RND(1)*62),RND(1)*15
190  NEXT I
200  CIRCLE(200,30),T,10:PAINT(200,30+T),10
210  FOR H=1 TO 11
220   LINE(0,63+H*H)-(256,63+H*H),3
230  NEXT H
240  DATA 170,142,121,104,90,78,70
250  F=T/3+1:PLAY"O=F;CDEFG"
260 NEXT T
270 RUN"CAS:" (270 GOTO 270)
```

Dat LINE-opdrachten ijzersterk zijn, wordt door dit programma wel bewezen. Boven deze voorbijtrekkende asteroïde zou je zo een UFO kunnen verwachten. Het hoofdstuk dat sprites beschrijft, zal dat dan ook mogelijk maken. De LINE-opdracht is zeer veelzijdig.

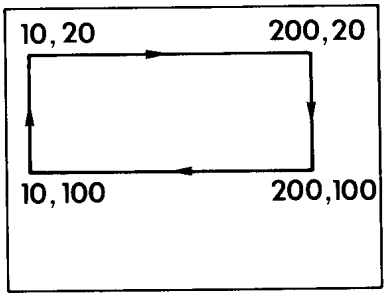
```
LINE(10,20)-(200,100),,B
```

trekt een lijn van beeldpunt (10,20) naar beeldpunt (200,100).

```
LINE(10,20)-(200,100),6
```

doet datzelfde in de kleur rood.
Met de LINE-opdracht zoals die hierboven staat, kunnen we een rode rechthoek (zie afbeelding 8-1) als volgt tekenen:

```
LINE(10,20)-(200,20),6
LINE(200,20)-(200,100),6
LINE(200,100)-(10,100),6
LINE(10,100)-(10,20),6
```



Afbeelding 8-1

Deze procedure is tamelijk omslachtig, vraagt veel typewerk en geeft dus veel kans op fouten.
De LINE-opdracht heeft een paar specialiteiten om het tekenen van rechthoeken te vereenvoudigen. Een doosje heet in het Engels 'box'. Door aan de LINE-opdracht de eerste letter van deze Engelse benaming (B) toe te voegen, hoe ven we slechts twee hoekpunten van een rechthoek op te geven om hem keurig op het scherm te krijgen:

```
LINE(10,20)-(200,100) ,6, B
```

Het kan natuurlijk voorkomen dat we een rechthoek in de op een bepaald ogenblik geldende inktkleur willen hebben. Het kleurcijfer mag dan vervallen, echter niet de bijbehorende komma!

```
LINE(10,20)-(200,100), ,B
```

Deze regel geldt voor veel opdrachten waarin een aantal variabelen kan worden vastgelegd. Wordt in dit geval de komma per ongeluk weggelaten dan staat er

```
LINE(10,20)-(200,100) ,B
```

Dat is geen rechthoek, maar een diagonale lijn in de kleur zwart (B=0) of een andere kleur als de B in het programma al een andere waarde had. MSX-BASIC kent de PAINT-opdracht. Bijvoorbeeld:

```
LINE(10,20)-(200,100),6,B:PAINT(50,50),6
```

Tussen de haakjes van de PAINT-opdracht wordt een punt opgegeven dat binnen de in te kleuren rechthoek valt. Bij het gebruik van PAINT wordt achter de haakjes het kleurnummer opgegeven. Dit moet dezelfde kleur zijn als die waarmee de rechthoek is getekend. Mocht dat niet het geval zijn, dan kookt de kleur over en vult het hele scherm.
Ook voor het inkleuren van rechthoeken heeft MSX-computer een foefje ingebouwd. De twee hiervoor afgedrukte opdrachten kunnen door een enkele worden vervangen:

```
LINE(10,20)-(200,100),6,BF
```

De toevoeging F is de eerste letter van het Engelse woord 'fill', dat vullen betekent.
Alvorens de LINE-opdracht verder te bekijken, rijgen we even twee pareltjes aan de ketting. Pareltjes die gebruik maken van de LINE-opdrachten zoals die hiervoor beschreven zijn.

```
10 REM rechthoeken
20 SCREEN 2:COLOR,1,1:CLS
30 FOR I=1 TO 100
40   LINE(0,0)-(RND(1)*256,RND(1)*192),
     RND(1)*16,BF
50   LINE(256,0)-(RND(1)*256,RND(1)*192) ,
     RND(1)*16,BF
```

```
60   LINE(256,192)-(RND(1)*256,RND(1)*192),
     RND(1)*16,BF
70   LINE(0,192)-(RND(1)*256,RND(1)*192),
     RND(1)*16,BF
80  NEXT  I
90  RUN"CAS:" (90   GOTO   90)
```

Vanuit de vier hoeken worden in een razend tempo rechthoeken getekend van willekeurige afmetingen in een willekeurige kleur. Jammer dat Pieter Mondriaan met dit soort technieken al furore maakte, anders had er zeker brood in gezeten. Het volgende programma maakt de schilderijtentoonstelling compleet. Hier wordt uitbundig gebruik gemaakt van de invulopdracht. De snelheid waarmee de computer zijn kleurboek vult, is bepaald indrukwekkend.

```
10 REM ingekleurd spectakel
20 SCREEN 2:COLOR ,4,4:CLS
30 FOR I=1 TO 100
40   LINE(RND(1)*256,RND(1)*192)-
      (RDN(1)*256,RND(1)*192),RND(1)*16,BF
50 NEXT I
60 RUN"CAS:" (60 GOTO 60)
```

De LINE-opdracht kent nog meer mogelijkheden. Voor het trekken van een lijn vanaf het beeldpunt dat het laatst door de computer is getekend naar een ander punt, laten we het eerste deel van de opdracht weg.

```
LINE -(100,150)
```

De mogelijkheid om met kleur gevulde vierkanten te maken, blijft bestaan.

```
LINE  -(100,150),10,BF
```

Een voorbeeld waarin de verkorte LINE-opdracht is toegepast, wordt gegeven in het volgende programma. Door het gebruik van een goniometrische uitdrukking wordt een spiraal op het scherm getekend.

```
10  REM wentelgoot
20  SCREEN 2:COLOR 11,4,4:CLS
30  A=30:B=0:PI=3.14
40  PSET(180,0)
50  FOR T=0 TO 180 STEP PI/25
60    X=50*COS(T):G=126+X
70    A=A+1:B=B+1
80    LINE -(G,B)
90    LINE(126,A)-(G,B)
100 NEXT T
110 RUN"CAS:"   (110 GOTO 110)
```

In een ander voorbeeld wordt de verkorte LINE-opdracht gebruikt om een driehoek te tekenen.

```
10 REM driehoek met bal
20  SCREEN 2:COLOR 11,4,4:CLS
30  Y=192
40  FOR X=0 TO 192
50   PSET(O,O)
60   LINE-(X,Y)
70  NEXT X
80  FOR I=0 TO 190
90   CIRCLE(I,I-9),5,4
100   CIRCLE(I+1,I-8),5,15
110 NEXT I
120 RUN"CAS:"  (120 GOTO 120)
```

Als de bal langs de driehoek naar beneden rolt, geeft hij een prachtige demonstratie van het effect dat door de beeldpuntgroepjes van acht wordt veroorzaakt. Er is met het beeld nog iets vreemds aan de hand. Omdat het aantal beeldpunten op de horizontale as gelijk is aan het aantal beeldpunten op de verticale as (192), zouden we een diagonaal doorgesneden vierkant mogen verwachten. Om het nog duidelijker te maken, typen we de volgende regels in:

```
10  SCREEN  2
20  LINE   (0,0)-(100,100),2,BF
30  GOTO  30
```

Dit zou een vierkant op moeten leveren met zijden van 100 beeldpunten. Het vierkant is echter geen vierkant maar een rechthoek!
Herinnert u zich de cirkel nog die de toevoeging 1.35 nodig had om rond te worden? Dit verschijnsel doet zich hier ook voor (en ook bij gebruik van het lageresolutiescherm 3). Het is een nogal storende slordigheid in het MSX-systeem.
Er zijn twee mogelijkheden om dit euvel te verhelpen.

- Bij alle tekenopdrachten wordt de Y-coördinaat vermenigvuldigd met een factor 1.35.
- U verdraait de beeldhoogte-instelling van de TV (zit meestal achterop het toestel) tot het grafische scherm beeldvullend is. De onder- en boven-rand verdwijnen dan. Deze afregeling kan het beste gebeuren met de rechthoek uit het voorgaande programma in beeld. Dan is goed te beoordelen of de afregeling het gewenste resultaat heeft.

In dit boek is gekozen voor de eerste methode. Het is namelijk zeer onpraktisch om bij de wisseling van een tekstscherm naar een grafisch scherm de beeld-hoogte opnieuw af te regelen. Als het toestel bovendien ook nog gebruikt wordt als TV-ontvanger, dan kunt u de grootste problemen met eventuele medegebruikers verwachten. Stel je voor, een weerman meteen lang gezicht. Dat kan niet, ook al is er vaak een goede reden voor.

Een laatste variant van de LINE-opdracht is de stap-functie.

```
LINE(20,70)-STEP(30,10)
```
of:

LINE -STEP(30,10)

In het eerste geval wordt er een lijn getrokken vanaf beeldpunt (20,70) waarbij het eindpunt 30 beeldpunten naar rechts en 10 beeldpunten naar beneden komt te liggen. De tweede opdracht doet hetzelfde vanaf het beeldpunt dat het laatst door de computer is opgeroepen.Hiermee hebben wede LINE-opdracht voldoende onder de loep genomen om er goed mee overweg te kunnen.
Ter afsluiting van dit hoofdstuk nog twee lijnenspelletjes.

```
10 REM verlichte stad
20 SCREEN 2:COLOR,1,1:CLS
30 PSET (120,60)
40 FOR I=1 TO 1000
50   X=RND(l)*10:Y=RND(l)*6:C=RND(l)*14
60   LINE -STEP(SGN(RND(l)-.5)*
      X,SGN(RND(1)-.5)*Y),C
70 NEXT I
80 RUN"CAS:" (80 GOTO 80)
```

Wat het voorstelt, is niet helemaal duidelijk: een landkaart, een asteroïdengordel, of een stad bij nacht? De laatste regel geeft alvast aansluiting met een volgende parel. In het allerlaatste pareltje van dit hoofdstuk lopen we alvast wat vooruit.

```
10 REM servetring
20  SCREEN 2:COLOR 10,4,10:CLS
30  FOR HOEK=.5*3.14 TO 1.5*3.14 STEP .1
40   GOSUB 100
50  NEXT HOEK
60  FOR HOEK=1.5*3.14 TO 2.5*3.14 STEP .03
70   GOSUB 100
80  NEXT HOEK
90  RUN"CAS:" (90 GOTO 90)
100 X=40*SIN(HOEK):Y=40*COS(HOEK)
110 PSET(130+X,90+Y)
120 LINE -STEP(0,50)
130 RETURN
```

Met behulp van een goniometrisch sommetje in regel 100 worden de punten van een cirkel berekend. Vanaf deze punten wordt een verticaal lijnstukje getrokken. Het programma laat zien dat het maken van berekeningen in BASIC erg traag gaat. In het volgende hoofdstuk worden de cirkels dan ook niet berekend. We gaan daar wat dieper in op het gebruik van de CIRCLE-opdracht, één van de mooiste grafische opdrachten die de MSX-computer kent.

9 Cirkels en aanverwante zaken

We zijn de cirkel-opdracht in dit boek al vele malen tegengekomen. In dit hoofdstuk blijven we er even bij stilstaan. De cirkel is een buitengewoon belangrijke vorm als we grafisch met de MSX-computer willen werken. Zoals we intussen gewend zijn, vallen we meteen met de deur in huis. We geven de computer geen tijd om af te koelen en typen in:

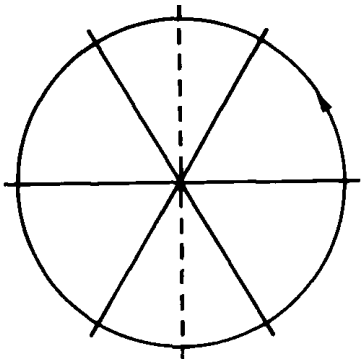
```
10 REM cirkels
20 SCREEN 2:COLOR 15,4,4:CLS
30 FOR I=1 TO 100
40 CIRCLE(RND(1)*256,RND(1)*192)
 RND(1)*100
50 NEXT I
60 RUN"CAS:"          (60 GOTO 60)
```

We hadden al geconstateerd dat de op deze wijze verkregen cirkels niet rond zijn. Toch wel een mooi gezicht. Wellicht een goed idee voor behangpapier. Hoe de cirkels rond worden, is inmiddels ook duidelijk.

```
10 REM ronde cirkels
20 SCREEN 2:COLOR ,l,l:CLS
30 FOR I=l TO 100
40   CIRCLE(RND(1)*256,RND(1)*192),
RND(1)*100,RND(1)*14,,,1.35
50 NEXT I
60 RUN"CAS:"         (60 GOTO 60)
```

In vergelijking met de CIRCLE-opdracht in het voorafgaande programma zijn er een paar toevoegingen. De beeldpuntcoördinaten tussen de haakjes en de straal direct achter de haakjes zijn onveranderd gebleven. Achter de straal is de gewenste kleur opgegeven. Vervolgens zijn er twee posities opengelaten. Op deze plaatsen kunnen de begin- en eindhoek worden ingevuld als we slechts een deel van de cirkel willen tekenen. Zorg ervoor geen enkele komma te vergeten. Als sluiter van de rij is het getal opgegeven dat de rondheid van de cirkel vastlegt. Dat wil zeggen de verhouding tussen de horizontale en de verticale as. Om op een normaal TV-beeldscherm een ronde cirkel te krijgen, moeten we, zoals we hebben gezien, hier 1.35 invullen.
Om het gebruik van de CIRCLE-opdracht ook voor de niet-wiskundige een beetje toegankelijk te maken, voeren we de appeltaart ten tonele. Een cirkel wordt linksom getekend, waarbij als eerste punt het beeldpunt rechts van het middelpunt wordt getekend. Tenminste, als we er verder niets aan doen. Als een cirkel volledig getekend is, heeft de straal van de cirkel zich

over een hoek van 2*pi radialen bewogen. Als er slechts een deel van de cirkel hoeft te worden getekend, dan kan de programmeur met een wiskundige achtergrond de begin-en eindhoek opgeven in radialen. Dooreen beetje te smokkelen met de nauwkeurigheid van de hoeken kunnen we de wiskunde vergeten en de cirkel als een appeltaart beschouwen. Afbeelding 9-1 laat een royale zespersoons appeltaart zien.



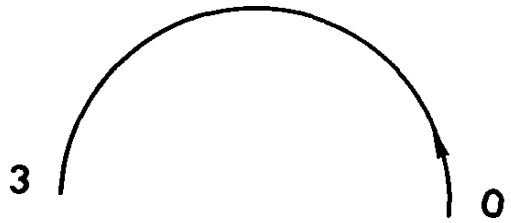
Afbeelding 9-1 Een in zes stukken verdeelde cirkel.

We verdelen de taart in zes punten. De hoekpunten van de stukken nummeren we van O tot en met 6. Willen we de cirkelboog van taartpunt 0-1 tekenen, dan vullen we eenvoudig die twee getallen in. Zo dus:

```
CIRCLE(1ØØ,1ØØ),3Ø,6,Ø,1,1.35
```

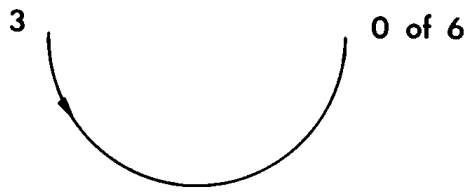
x y r kleur beginhoek eindhoek rondheid

De bovenste helft van een cirkel ziet er dan uit zoals in afbeelding 9-2.



Afbeelding 9-2

De cirkelboog loopt immers van 0 tot 3. De onderste helft van de cirkel staat in afbeelding 9-3.



Afbeelding 9-3

```
CIRCLE(100,100),30,6,3,6,1.35
```

De boog loopt hier van 3 tot 6. Met even veel recht kunnen we zeggen dat de cirkelboog van 3 naar 0 loopt.

```
CIRCLE(100,100),30,6,3,0,1.35
```

Het resultaat op het scherm is dan ook hetzelfde. Hiermee is dan meteen aangegeven hoe cirkeldelen in de rechterhelft worden opgegeven. We gaan gewoon door de 0 heen, waarbij we altijd linksom draaien. Van 5 tot 1 geeft dus een derde cirkelboog met de bolle kant naar rechts. In de notering mogen ook cijfers achter de komma (voor de computer is dat dus een punt) voorkomen. Dat is vaak handig als we ergens precies op aan willen sluiten. Door gebruik te maken van de verdeling van - tot 6 maken we een kleine fout. Voor een volledige cirkel zou het laatste getal 6.28 moeten zijn. Dat is eenvoudig te controleren door twee cirkels te tekenen. Eén van 0 tot 6 en één van 0 tot 6.28.

```
10 REM cirkelbenadering
20 SCREEN 2:COLOR ,4,4:CLS
30 CIRCLE (70,90),50,15,0,6,1.35
40 CIRCLE (170,90),50,15,0,6.28,1.35
50 GOTO 50
```

Omdat we de begin- en eindhoek alleen maar opgeven als we cirkeldelen willen tekenen, is het gemaakte foutje bij de verdeling van 0 tot 6 zonder meer acceptabel. Zeker gezien het gemak dat het gebruik van ronde getallen met zich meebrengt. Het voorgaande programma is leerzaam, maar niet interessant genoeg om er een pareltje van te maken. Daarom rijgen we, voor we gaan oefenen met delen van cirkels, even een ander pareltje aan het snoer.

```
10 REM volle bollen
20 SCREEN 2:COLOR,4,4:CLS
30 FOR 1=1 TO 30
40   X=RND(1)*256:Y=RND(1)*192:
       R=RND(1)*70:C=RND(1)*16
50   CIRCLE(X,Y),R,C,,,1.35
60   PAINT(X,Y),C
70 NEXT I
80 RUN"CAS:"       (80 GOTO 80)
```

We hebben aan onze ketting al een hele serie kleine pareltjes. Natuurlijke parels groeien in een oester. Ze zijn opgebouwd uit laagjes. Als we een parel langer laten groeien, kan er een prachtig groot exemplaar ontstaan. In het nu volgende deel van dit hoofdstuk gaan we laagsgewijs een grote parel opbouwen. Het wordt een juweeltje van een MSX-grafiek, voor een groot deel opgebouwd met de CIRCLE-opdracht.
Het is een mooie zomerse dag. De hemel is gevuld met kleine schapewolkjes.

```
10 REM schapewolkjes
20 SCREEN 2,2:COLOR 15,4,4:CLS
30 LINE(0,120)-(256,192),3,BF
40 FOR I=1 TO 150
50   X=RND(1)*256:Y=RND(1)*30:R=RND(1)*10
60   CIRCLE(X,Y),R,,3.9,5.5
70 NEXT I
80 GOTO 80
```

Dat ziet er al aardig uit. De cirkelbogen lopen van 3.9 tot 5.5 terwijl ze iets afgeplat zijn, omdat de toevoeging 1.35 weggelaten is. Ook de kleuraanduiding is weggelaten, zodat ze de inktkleur wit (15) uit regel 20 hebben. Het getal pi komt er niet aan te pas.
Zou het niet mooier zijn als enkele speelse elementen het nogal saaie landschap zouden opfleuren? Wat boompjes aan de horizon, een landweggetje, wat begroeiing hier en daar?
Zo'n uitbreiding laat zich, met de kennis die we nu van cirkels hebben, betrekkelijk eenvoudig aanbrengen.

```
10  REM landschap
20  SCREEN 2:COLOR 15,4,4
30  LINE(0,120)-(256,192),3,BF
40  FOR I=1 TO 150
50    X=RND(1)*256:Y=RND(1)*30:R=RND(1)*10
60    CIRCLE(X,Y),R,,3.9,5.5
70    CIRCLE(X,120),R,12,,3
80  NEXT I
90  COLOR 10
100 PSET(135,120)
130 DRAW"M160,192M100,192M135, 120"
140 PAINT(135,125),10
150 FOR I=1 TO 10
160   CIRCLE(130-I*I/1.5,110+I*I/1.5),
         I*I/5,12,,,2
170 INT(130-I*I/1. 5,110 + I*I/1.5) , 12
180   LINE(130-I*I/1.5,110+I*I/1. 5)-
        (131-I*I/1.5,110 + I*I),12 ,BF
190   CIRCLE(140 + I*I/1.5,110 + I*I/1.5) ,
        I*I/5,12,,,2
200   PAINT(140 + I*I/1.5,110 + I*I/1.5) ,12
```

```
210   LINE(140 + I*I/1.5,110 + I*I/1. 5)-
         (141+I*I/1.5,110+I*I),12,BF
220 NEXT I
230 GOTO 230
```

Laat u niet ontmoedigen door de schijnbaar ingewikkelde CIRCLE-,PAINT- en LINE-opdrachten. Ze lijken erg veel op elkaar en door het veranderen van het regelnummer en enkele details kunnen ze van elkaar worden gekopieerd. De hier gebruikte DRAW-opdracht is een hoofdstukje apart waard. Dat volgt dan ook nog. Als het programma zijn werk gedaan heeft, zult u verbaasd staan over de beheersing die we al over de grafische capaciteiten van de MSX-computer hebben. Wie zou er in zo'n vredig landschap niet een wandeling willen maken. Toch moet u iets opvallen. Het standpunt van waaruit u het landschap bekijkt, is nogal hoog. Zou het soms kunnen zijn dat u dit landschap vanuit een hoog flatgebouw bekijkt? Bovendien is er nog heel wat anders aan de hand! Voeg de volgende regels maar eens toe:

```
222 FOR I=0 TO 256
224   PSET (I,60),14
226  FOR Q=l TO 40:NEXT Q
228 NEXT I
```

Probeer voor het intypen van RUN eens te raden wat er zal gebeuren door deze toevoeging. Alleen het geluid ontbreekt nog.
Sporen van verkeersvliegtuigen worden gevormd door gecondenseerd, soms zelfs bevroren, water dat zich in het uitlaatgas bevindt (bij verbranding van kerosine komt water vrij). Vaak is, zoals in ons programma, het vliegtuig zelf niet eens te zien.
Toch zou het aardig zijn te kunnen beschikken over een echt vliegtuig. We gaan er een bouwen met behulp van een sprite. Hoe dat precies in zijn werk gaat, onderzoeken we in een volgend hoofdstuk. We typen, zonder er al te veel bij stil te staan, een aanvullend stukje programma. Voor we dat doen, verwijderen we de zojuist toegevoegde regels 222 tot en met 228.

```
10 REM  landschap
20 SCREEN  2,2:COLOR  15,4,4
30 LINE(0,120)-(256,192),3,BF
40 FOR  I=1   TO  150
50 X=RND(1)*256:Y=RND(1)*30:R=RND(1)*10
60 CIRCLE(X,Y),R,,3.9,5.5
70 CIRCLE(X,120),R,12,,3
80 NEXT I
90 COLOR  10
100 PSET(135,120)
130 DRAW"M160, 192M100,192M135,120"
140 PAINT(135,125),10
150 FOR  I=1 TO 10
160 CIRCLE(130-I*I/1.5,110 + I*I/1. 5),
    I*I/5,12,,,2
```

```
170 PAINT(130-I*I/1.5,110 + I*I/1. 5) ,12
180 LINE(130-I*I/1.5,110+I*I/1.5)-
   (131-I*I/1.5,110+I*I),12,BF
190 CIRCLE(140 + I*I/1.5,110 + I*I/1.5) ,
    I*I/5,12,,,2

200   PAINT(140 + I*I/1.5,110 + I*I/1. 5) , 12
210   LINE(140+I*I/1.5,110+I*I/1.5)-
(141+I*I/1.5,110+I*I),12,BF
220 NEXT I
```

Dit hadden we dus al. Let echter goed op. In regel 20 staat: SCREEN 2,2. We vertellen de computer hiermee dat we een grote sprite van 16 bij 16 beeldpunten in gaan voeren. De rest van het programma is nog niet eerder ingetypt.

```
230 DATA 0,0,14,133,194,239,112,64,254,194,
      132,9,14,0,0,0
240 DATA 0,0,0,0,128,252,2,1,127,64,128,0,
      0,0,0,0
250 FOR S=l TO 32
260 READ A
270   S$=S$+CHR$(A)
280 NEXT S
290 SPRITE$(0)=S$
300 SOUND 7,55
310 SOUND 8,16
320 SOUND ll,255:SOUND 12,120:SOUND 13,13
330 FOR I=0 TO 255
340   SOUND 6,INT(I/9)
350   IF I=170 THEN SOUND 13,0
360   PUT SPRITE 0,(I,52),15,0
370   PSET(I-10,60),15
380   PSET(I-50,60),14
390  PSET(I-90,60),4
400 NEXT I
410 RUN"CAS:" (410 GOTO 410)
```

Dit ruw verstoord landelijk tafereeltje is voor een parel nogal fors. Toch is het verwonderlijk dat we in minder dan veertig regeltjes bijzonder mooie achtergronden kunnen maken waarin we elementen kunnen laten bewegen. Enkele schapen in de wei, een vlindertje in de lucht en meer van dat moois wordt pas echt goed mogelijk als we het ontwerpen van en manipuleren met sprites onderde knie hebben. Daar besteden we dan ook een uitgebreid hoofdstuk aan. Om de (rijg)draad van dit boek weer op te pakken, spelen we nog even met CIRCLE, zonder een andere bedoeling dan zomaar wat aardige prentjes te maken.
We hebben al eens eerder een logo van een groot bedrijf gemaakt op het lage-resolutiescherm. In hoog oplossend vermogen (SCREEN2), ziet dat er zó uit:

```
10 REM logo
20 SCREEN 2:COLOR 10,4,4:CLS
30 OPEN"GRP:" FOR OUTPUT AS #1
40 PSET(103,87),4
50 PRINT#1,"DuPont"
60 CIRCLE(124,90),30,,,,.5
70 RUN"CAS:"      (70 GOTO 70)
```

Let weer goed op de punten en de komma's. Kent u het volgende verkeersbord?

```
10 REM verkeer
20 SCREEN 2:COLOR ,l,l:CLS
30 OPEN"GRP:" FOR OUTPUT AS #1
40 CIRCLE(130,100),90,6,,,1.35
50 PAINT(130,100),6
60 CIRCLE(130,100),70,15,,,1.35
70 PAINT(130,100),15
80 PSET(0,0),1
90 PRINT#l,"VERKEER"
100 RUN"CAS:"     (100 GOTO 100)
```

Nu draaien we alle kleurnummers om.

```
10 REM soep
20 SCREEN 2:COLOR ,l,l:CLS
30 OPEN"GRP:" FOR OUTPUT AS #1
40 CIRCLE(130,100),90,15,,,1.35
50 PAINT(130,100),15
60 CIRCLE(130,100),70,6,,,1.35
70 PAINT(130,100),6
80 PSET(0,0),1
90 PRINT#1,"TOMATENSOEP"
100 RUN"CAS:"     (100 GOTO 100)
```

Een oud grapje, dat wel. Maar toch mooi geïllustreerd. Kent u het volgende bord?

```
10 REM verkeer
20 SCREEN 2=COLOR ,15,15:CLS
30 OPEN"GRP:" FOR OUTPUT AS #1
40 CIRCLE(130,100),90,6, , ,1.35
60 CIRCLE(130,100),70,6, , ,1.35
70 PAINT(130,171),6
80 LINE(80,125)-(180,45),6
90 LINE(85,145)-(185,65),6
100 PAINT(128,96),6
110 RUN"CAS:"    (110 GOTO 110)
```

In deze miniprogramma's komt de PAINT-opdrachtveelvuldig voor. We zullen er op de nu volgende pagina's een hoofdstukje aan wagen.

10 De schilderskwast ter hand genomen

Met de PAINT-opdracht kunnen we vlakken schilderen. Die vlakken mogen alle vormen aannemen. Er zijn echter drie voorwaarden waaraan moet worden voldaan:
De contouren die het in te kleuren vlak bepalen, moeten volledig gesloten zijn.
De lijn waarmee de contour is getekend, moet de kleur hebben waarmee wordt ingekleurd.
De coördinaten in de PAINT-opdracht moeten binnen het vlak vallen en mogen niet op de buitenste rand van beeldpunten vallen. Laten we met een voorbeeld beginnen:

```
10 REM glas
20 SCREEN 2:COLOR 15,15,15:CLS
30 LINE(0,130)-(256,192),12,BF
40 CIRCLE(130,40),35,6,,,.3
50 CIRCLE(130,170),26,6,,,.3
60 CIRCLE(130,100),31,6,,,.3
70 LINE(94,40)-(104,170),6
80 LINE(165,40)-(156,170),6
90 GOTO 90
```

De bovenstaande regels zorgen ervoor dat er een realistisch getekend glas op tafel komt te staan. Stel dat we dit glas willen vullen met tomatensap, dan moeten we drie PAINT-opdrachten geven omdat er drie gesloten contouren aanwezig zijn die moeten worden gevuld.

```
52 PAINT(130,170),6 62 PAINT(130,100),6
82 PAINT(130,135),6
```

Het kan ook anders. Als we een gaatje prikken in de twee ellipsen die de begrenzing van de vloeistof aangeven, dan kunnen we met slechts één PAINT-opdracht volstaan. De verf lekt dan door die gaatjes vanzelf in de andere delen van de tekening. Dat doorprikken van een ellips gaat het best met een naald in de achtergrondkleur. Omdat ertwee achtergrondkleuren zijn, hebben we twee naaldjes nodig. We halen de regels 52, 62 en 82 weer weg en voegen de naalden toe.

```
82 LINE(130,100)-(130,130),15
84 LINE(130,130)-(130,170),12
86 PAINT(130,177),6
88 CIRCLE(130,100),31,9,,,.3
```

Het glas wordt nu gevuld door één enkele PAINT-opdracht. Dat laatste lichtrode randje maakt er een smakelijk uitziend drankje van. De uitgelopen groepjes van acht beeldpunten geven er zelfs iets levendigs aan. Na RENUM en de toevoeging van de koppeling met een volgende parel ziet de 'Bloody Mary' er zó uit:

```
10 REM glas
20 SCREEN 2:COLOR 15,15,15:CLS
30 LINE(0,130)-(256,192),12,BF
40 CIRCLE(130,40),35,6,,,.3
50 CIRCLE(130,170),26,6,,,.3
60 CIRCLE(130,100),31,6,,,.3
70 LINE(94,40)-(104,170),6
80 LINE(165,40)-(156,170),6
90  LINE(130,100)-(130,130),15
100 LINE(130,130)-(130,170),12
110 PAINT(130,177),6
120 CIRCLE(130,100),31,9,,,.3
130 OPEN"GRP:" FOR OUTPUT AS #1
140 PSET(0,10),15
150 COLOR 1
160 PRINT#1,"GEZONDHEID"
170 RUN"CAS:"    (170 GOTO 170)
```

Voor de afwisseling rijgen we een fantasieprogramma aan de draad, waarin PAINT een hoofdrol speelt.

```
10 REM papier
20 SCREEN 2:COLOR ,4,4:CLS
30 FOR I=1 TO 5
40 COLOR I+8
50 PSET(100,170-30*1)
60 LINE -STEP(-50,50)
70 LINE -STEP(100,0)
80 LINE -STEP(50,-50)
90 LINE -STEP(-100,0)
100 PAINT(110,180-30*I)
110 NEXT I
120 RUN"CAS:" (120 GOTO 120)
```

Dit programma demonstreert weer eens de ongekende mogelijkheden van de MSX-computer op het gebied van grafische weergave. Let eens op het nuanceverschil tussen geel en lichtgeel. Verander nu de kleurnummers4 in regel 20 in 1. De achtergrond is nu zwart in plaats van blauw. Het nuanceverschil tussen geel en lichtgeel is nu grotendeels verdwenen. Bij gebruik van een duurdere monitor treden dit soort kleurverschuivingen niet op. Als homecomputerhobbyisten zullen we er mee moeten leren leven.
Met kleine veranderingen in het bovenstaande programma zijn prachtige resultaten te behalen.

```
10 REM nog meer papier
20  SCREEN 2:COLOR ,4,4:CLS
30  FOR I=2 TO 15
40   COLOR I
50   PSET*100, 150-10*I)
60   LINE -STEP(-50,50)
70   LINE -STEP(100,0)
80   LINE -STEP(50,-50)
90   LINE -STEP(-100,0)
100  PAINT(110,180-10*I)
110 NEXT I
120 RUN-CAS:" (120 GOTO 120)
```

Een pak vouwblaadjes voor de kleuterschool.

In het lageresolutiescherm (3) kan aan PAINT nog een opdracht worden toegevoegd. Behalve de X- en Y-coördinaat en de verfkleur, kan ook de kleur van de omtrek worden opgegeven. Heeft u daar behoefte aan? We maken op de PAINT-valreep gauw nog een pareltje voor de ketting. De stapel papier wordt op kleurvolgorde gelegd.

```
10 REM nog veel meer papier
20  SCREEN 2:COLOR ,1,1:CLS
30  FOR I=1 TO 14
40    READ C:COLOR C
50   PSET(100, 150-10*I)
60   LINE -STEP(-50,50)
70   LINE -STEP(100,0)
80   LINE -STEP(50,-50)
90   LINE -STEP(-100,0)
100  PAINT(110,180-10*I)
110 NEXT I
120 DATA 12,2,3,9,8,6,13,4,5,7,14,15,10,11
130 RUN"CAS:" (130 GOTO 130)
```

Ten opzichte van de vorige stapel zijn de regels 30 en 40 aan verandering onderhevig geweest. Bovendien zijn de kleurnummers in een DAT A-regel opgenomen. Het kleurenpalet van de MSX-computer is heel bruikbaar, al laat het hier en daar toch wel te wensen over. Zo ontbreekt de kleur oranje. In het volgende hoofdstuk proberen we zelf oranje en andere mengkleuren te maken.

11 Een cursus verf mengen

Om de kleur oranje te maken, moeten we op de een of andere wijze de kleuren rood en geel in een bakje doen en flink roeren.
Op het beeldscherm is de beeldpunt het kleinste element. We kunnen proberen oranje te maken door de beeldpunten om en om rood en geel te maken. De mengkleur van rood en geel is immers oranje. Het volgende programma laat zien hoe dat werkt.

```
10 REM oranje
20 SCREEN 2:COLOR ,6,6:CLS
30 FOR Y= 50 TO 100
40   FOR X=0 TO 256 STEP 2
50    PSET(X,Y),10
60 NEXT X,Y
70 GOTO 70
```

Dat werkt dus niet. Natuurlijk niet, zult u denken.
Door de eerder behandelde horizontale groepjes van acht beeldpunten moeten we als achtergrondkleur één van de te mengen kleuren nemen. Er kunnen dan, per groepje van acht horizontale beeldpunten, nooit meer dan twee kleuren voorkomen.

```
10 REM oranje?
20 SCREEN 2:COLOR ,4,4:CLS
30 FOR Y=50 TO 100
40  FOR X=0 TO 256 STEP 2
50     PSET(X,Y),10:PSET(X,Y),6
60 NEXT X,Y
70 GOTO 10
```

Het kan dus toch! Oranje op de buis. Ondanks dat de opbouw van de kleur nogal traag gaat, kunnen we de kleur oranje die we zojuist hebben gemaakt toepassen in programma's.
Een klein programma bewijst weer eens dat er meer in de MSX-computer zit dan er in de handboeken staat.

```
10 REM hoera
20 SCREEN 2:COLOR ,6,3:CLS
30 FOR Y=40 TO 60
40 FOR X=48 TO 150 STEP 2
50 PSET(X,Y),10
60 NEXT X,Y
70 LINE(48,70)-(150,90),6,BF
80 LINE(48,90)-(150,110),15,BF
```

```
90  LINE(48,110)-(150,130),4,BF
100 LINE(47,39)-(152,61),3,B
110 LINE(47,69)-(151,131),3,B
120 PAINT(1,1),3
130 LINE(44,30)-(47,180),1,B
140 RUN"CAS:"    (140 GOTO 140)
```

Dooreen truc uit te halen, kunnen we dus toch mengkleuren maken tegen elke gewenste achtergrondkleur, die zelf ook weer een mengkleur kan zijn. De mengkleur wordt afgeschermd met een lijntje in de gewenste achtergrond-kleur (de regels 100 en 110) en een PAINT-opdracht doet de rest. De coördinaten van de PAINT-opdracht moeten dan natuurlijk buiten de reeds ingekleurde rechthoeken liggen. Om onze artistieke talenten op de computer uit te kunnen leven, zullen we eens wat kleurtjes mengen. Uit een stuk triplex zagen we een palet, waarop we de te mengen kleuren uitspreiden. Door het palet in de kleur te schilderen waarmee we enkele andere kleuren willen mengen, kan het programma relatief eenvoudig blijven. Penseel bij de hand?

```
10  REM palet
20  SCREEN 2:COLOR 7,7,1:CLS
30  FOR Y=10 TO 70
40   FOR X=16 TO 88 STEP 2
50     PSET(X,Y),4
60  NEXT X,Y
70  FOR Y=45 TO 105
80   FOR X=103 TO 175 STEP 2
90     PSET(X,Y),9
100 NEXT X,Y
110 FOR Y=130 TO 190
120   FOR X=160 TO 232 STEP 2
130    PSET(X,Y),3
140 NEXT X,Y
150 CIRCLE(50,192),200,1
160 PAINT(255,1),1
170 PSET(0,187)
180 LINE -STEP(70,-40),1
190 LINE -STEP(3,10),1
200 LINE -STEP(-50,34),1
210 PAINT(1,191),1
220 PSET(70,147),1
230 DRAW"U3E2U2E2U3E4U3E5R12F4R5F6"
240 DRAW"L6G9D2G4D3G3G1L6D1L6"
250 RUN"CAS:"        (250 GOTO 250)
```

Dat begint er op te lijken. Toch doen de vierkante verfvoorraadjes wat vreemd aan. Niet gek voor een kubist, maar Rembrandt zou er moeite mee hebben gehad. Bovendien kan er alleen maar worden gemengd met een enkele grond-kleur. Nu we de basisopzet voor dit programma eenmaal hebben, is het niet

moeilijk meer de scherpe kantjes er af te halen. Let goed op details zoals coördinaten en kleurnummers.

```
10 REM palet 2
20 SCREEN 2:COLOR 15,4,4:CLS
30 PSET(16,10):LINE -STEP(72,60),6,BF
40 PSET(103,45):LINE -STEP(72,60),13,BF
50 PSET(160,128):LINE -STEP(72,60),3,BF
60 FOR Y=10 TO 70
70 FOR X=16 TO 88 STEP 2
80 PSET(X,Y),10
90 NEXT X,Y
100 FOR Y=45 TO 105
110 FOR X=103 TO 175 STEP 2
120 PSET(X,Y),7
130 NEXT X,Y
140 FOR Y=130 TO 190
150 FOR X=160 TO 232 STEP 2
160 PSET(X,Y),15
170 NEXT X,Y
180 CIRCLE(50,40),30
190 CIRCLE(142,75),30
200 CIRCLE(195,159),30
210 CIRCLE(50,192),192
220 PAINT(1,191)
230 PSET(0,187)
240 LINE -STEP(70,-40),1
250 LINE -STEP(3,10),1
260 LINE -STEP(-50,34),1
270 PAINT(1,191),1
280 PSET(70,147),1
290 DRAW"U3E2U2E2U3E4U2E5R12F4R5F6"
300 DRAW"L6G9D2G4D3G3G1L6D1L6"
310 RUN"CAS:"        (310 GOTO 310)
```

De meeste scherpe kantjes zijn nu van de verfvlekken af. Toch speelt het regeltjevan acht ons weer parten. Omdat we niet kunnen vermijden dat er meer dan twee kleuren per acht horizontale beeldpunten voorkomen, zullen we met deze afwijking van het volmaakte moeten leren leven. Toch is het onverwacht dat we met de MSX-computer zelfs pasteltinten op het scherm kunnen toveren.
In het volgende hoofdstuk kijken we naar een opdracht waarvoor in MSX-BASIC een op zichzelf staand minitaaltje is ontwikkeld. Het is de DRAW-opdracht, een combinatie van de mogelijkheden van PSET, LINE en COLOR. We zijn deze opdracht al een paar keer tegengekomen.

12 Roept u maar!

De DRAW-opdracht luistert naar woorden. Het is een soort robot die zich op commando over het scherm beweegt. We roepen: "drie plaatsen naar links", en DRAW voert de opdracht die tussen aanhalingstekens staat keurig uit. In BASIC worden karakterreeksen strings genoemd; deze staan tussen aanhalingstekens. Een variabele die een string representeert, is te herkennen aan het dollar-teken ($) achter zijn naam. De uitroep "drie plaatsen naar links" is dus een string.
Speciaal voor de DRAW-opdracht is een taaltje ontwikkeld, een soort vakjargon, dat ons in staat stelt de robot naar elke plaats op het scherm te bewegen. Op die manier kunnen we een tekening maken.

```
10 REM DRAW
20 SCREEN 2:COLOR 15,4,4:CLS
30 PSET(100,50)
40  DRAW"R70D40L60U30R50D20L40U10R20 "
50  GOTO  50
```

In regel 40 is te zien dat we met een letter de richting en met een getal het aantal beeldpunten op kunnen geven. Zo betekent 'L40' ga veertig beeldpunten naar links. Omdat BASIC een internationaal gebruikte taal is, worden de richtingen in het Engels aangeduid.

Omhoog	Up	U
Omlaag	Down	D
Rechts	Right	R
Links	Left	L

Diagonaal bewegen kan ook. De ontwerper van de DRAW-subtaal is echter niet in staat geweest hier gemakkelijk te onthouden benamingen voor te verzinnen.

Rechts	Omhoog	E
Links	Omhoog	H
Rechts	Omlaag	F
Links	Omlaag	G

Behalve de één-assige verplaatsingsopdrachten kunnen ook coördinaten worden opgegeven. In dat geval gebruiken we de MOVE-opdracht.

M 10,30

trekt een lijn naar het punt met de coördinaten (10,30). Evenals bij de LINE-opdracht kan in MOVE ook een STEP-functie worden opgenomen. Bij MOVE kan dit ook afzonderlijk voor beide coördinaten.

M-10,30

trekt een lijntje 10 beeldpunten naar links en naar Y-coördinaat 30.

M10,+30

betekent een lijn naar X-coördinaat 10 en een verplaatsing van 30 beeldpunten omlaag. Zodra we een verplaatsing nodig hebben zonder dat er een lijn wordt getrokken, brengt de letter B uitkomst.

```
10 SCREEN 2
20 PSET(IOO.IOO)
30 DRAW"BM-50,50"
40 DRAW"C10R100"
50 GOTO 50
```

Kijk nog even naar regel 40. Daar staat een nieuw type opdracht. Met C kunnen we een kleur aangeven. In dit geval geel (10). Nog een paar mogelijkheden:

```
10   SCREEN   2:COLOR  15,4,4:CLS
20  A$="R8U6L8D6"
30  B$="L8D6R8U6"
40  DRAW"BM130,90"
50  FOR  1=1   TO  50
60       DRAW"S=I;XA$;XB$;"
70  NEXT  I
80  RUN"CAS:" (GOTO  80)
```

Een aardig effect. In de regels 20 en 30 worden de bewegingsopdrachten buiten DRAW alvast opgebouwd. Uiteraard in de vorm van een string. Regel 40 brengt de tekenpunt naar het midden van het scherm. We zouden hier evengoed een PSET-opdracht voor kunnen gebruiken. In regel 60 wordt de opdracht uitgevoerd. S geeft de schaalfactor aan waarmee de getallen in de bewegingsopdrachten worden vermenigvuldigd. Omdat I toeneemt, ontstaan er steeds grotere rechthoeken. Als we gegevens buiten de DRAW-opdracht naar binnen willen halen, kan dat door er een X voor te zetten. In feite kan elke DRAW-opdracht ook met LINE en PSET worden uitgevoerd. Met DRAW echter kan compacter worden geprogrammeerd.
Regel 90 maakt dit werkje tot een pareltje voor het al indrukwekkend lang geworden rijgsnoer. Neem af en toe de moeite het complete oeuvre eens terug te spelen. U zult getroffen worden door de variëteit aan beelden die al op het bandje staan. Dat is nog eens wat anders dan vakantiedia's. Schrik er niet voor terug ook anderen lastig te vallen met uw werkstuk. Het is beter van kwaliteit

dan menige TV-serie. Zelf ontworpen STER-spotjes doen het trouwens ook erg goed aan een parelketting.
Vanwege de beperking tot rechte lijnen zullen we verder niet stilstaan bij de DRAW-opdracht. Dit boek is niet bedoeld om alle onderwerpen uitputtend te behandelen. Tot nu toe hebben we verschillende soorten lijnen getekend, recht en geknikt, cirkelvormen en ellipsen. Toch is er een type lijn dat nog niet geheel uit de verf is gekomen: tekenlijn die willekeurige vormen kan volgen. Het palet uit het vorige hoofdstuk liet een penseel zien. De kwast ervan werd gevormd door zo'n grillig gevormde lijn. In dit geval werd die lijn opgebouwd met de DRAW-opdracht, hetgeen toch tamelijk moeilijk gaat. In het volgende hoofdstuk bekijken we een methode om willekeurige tekeningen te produceren. Prentjes bijvoorbeeld, die al op papier staan en waarvan we graag een kopie op het scherm willen hebben. Afbeeldingen van familieleden, een favoriete stripfiguur. Of gewoon de plattegrond van uw huis als basis voor een spannend avonturenspel.

13 Vrij tekenen

In alle voorafgaande hoofdstukken hebben we gebruik gemaakt van voorgeprogrammeerde MSX-functies. We hebben gezien dat er heel wat grafisch spektakel mee kan worden opgewekt. We zijn echter ook beperkingen tegengekomen. Zo liet het vorige hoofdstuk zien dat de DRAW-opdracht zijn naam niet echt waarmaakt. Erg ingewikkelde tekeningen zijn er niet mee te maken. In dit hoofdstuk gaan we een gereedschap ontwikkelen waarmee we elke tekening kunnen maken die uit lijnen is opgebouwd. Om de nieuwsgierigheid te prikkelen, volgt een programma om een tekening te maken. De tekening wordt gemaakt volgens de methode die in dit hoofdstuk staat beschreven. Schrik niet van de enorme lijst getallen in de DATA-regels. Het zijn, op enkele uitzonderingen na, paarsgewijs alle X- en Y-coördinaten waaruit de tekening bestaat. Normaal gesproken wordt deze getallenrij automatisch op een bandje gezet en er vanzelf weer afgelezen.
Het in dit hoofdstuk te ontwikkelen tekenprogramma is dan ook zeer gebruikersvriendelijk. Fouten kunnen worden hersteld en de resultaten kunnen erg aantrekkelijk zijn. Dat bewijst bijvoorbeeld de tekening die op het scherm verschijnt bij het uitvoeren van het volgende programma. Toegegeven, het plaatje is nogal pikant. Misschien wel te gewaagd voor een computerboek. Maar niemand dwingt u de indrukwekkende getallenrij in te voeren. Het risico is geheel aan u. Slimmerikken zullen wellicht voorzichtig proberen slechts enkele regeltjes data in te typen, om op die manier een idee te krijgen wat voor vlees we in de kuip hebben. Wees gewaarschuwd, u bent verloren!

```
10 REM Lorelei
20 SCREEN 2:COLOR 1,10,10:CLS
30 DIM X(200):DIM Y(200)
40 FOR D=l TO 14
50 READ Q
60 FOR I=1 TO Q
70 READ X(I),Y(I)
80 IF I<2 THEN 100
90 LINE(X(I-1),Y(I-1))-(X(I),Y(I))
100 NEXT I
110 NEXT D
120 RUN"CAS:"                (120 GOTO 120)
150 DATA 28,111,149,116,137,120,124,123,110,
         125,105,125,105,120,99,125,105,125,
         105,128,89,132,75,136,63
160 DATA 139,55,139,55,142,49,146,44,152,41,
         158,42,162,45,165,51,167,59,169,73,
         172,91,177,116,182,139,188,157,195,
         174,199,184
170 DATA 19,159,42,163,37,166,34,173,32,177,
         34,182,42,184,51,188,75,188,75,191,
         108,191,108,192,133,192,133,192,133,
```

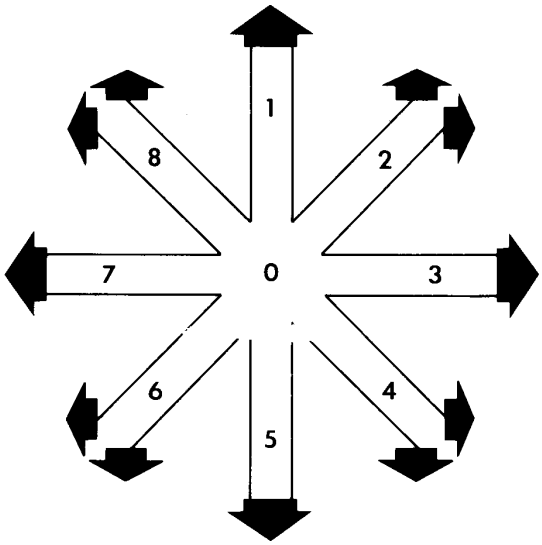
```
         194,157,196,164,200,174,201,180,204,
         185
180 DATA 19,147,64,148,67,147,73,147,73,147,
         79,148,91,150,102,153,113,155,118,
         162,135,169,150,174,163,174,169,174,
         169,173,174,173,182,173,182
190 DATA 176,189,176,189,99,154,117,154,130,
         153,147,152,157,151,162,150,163,150,
         169,150,173,149,177,146,183,141,186,
         133,186,121,183,117,180,117,180,111
200 DATA 174,108,170,104,161,100,153,94,146,
         88,140,80,133,71,122,65,112,69,102,
         66,111,63,118,63,118,63,124,62,132,
         54,134,61,132,68,137,71,138
210 DATA 75,138,81,138,87,142,93,151,102,172,
         02,172,102,174,101,174,94,160,100,
         175,99,177,97,176,91,162,96,179,94,
         178,93,177,87,163,90,177,89
220 DATA 178,87,175,84,167,84,164,81,161,79,
         155,77,152,77,152,69,153,59,152,59,
         152,52,149,48,144,48,163,48,126,48,
         126,49,119,49,114,49,114,50,103,51
230 DATA 93,51,93,53,85,53,85,57,77,62,72,67,
         70,67,70,72,62,72,62,74,64,70,51,70,
         51,70,48,70,48,66,34,68,28,73,21,74,
         19, 75, 19, 75, 19, 75, 14, 82,-16, 75, 6
240 DATA 75,6,68,4,66,7,21,68,9,62,5,62,5,58,
         7,51,14,51,14,46,54,46,24,46,33,46,
         37,48,40,48,47,52,60,57,78,59,72,62,
         69,64,67,64
250 DATA 65,64,65,63,61,63,61,11,169,150,168,
         157,165,165,161,173,157,176,151,177,
         150,179,151,169,153,163,161,153,159
260 DATA 13,79,16,80,18,81,19,81,20,80,21,79,
         22,77,21,82,24,83,25,83,26,82,28,81,
         29,81,29
270 DATA 61,81,21,84,24,85,26,86,30,86,32,84,
         2,83,33,82,35,82,36,81,36,84,37,85,
         36,86,34,86,33,89,37,89,39,89,41,89,
         41,86,45,84,46,83,47,79,48,75
280 DATA 48,84,46,84,46,84,52,84,55,85,58,89,
         60,95,60,95,60,10,61,105,63,107,66,
         107,66,109,72,109,74,106,74,109,74,
         112,76
290 DATA 114,76,119,79,120,80,120,80,120,81,
         122,80,122,80,121,82,122,84,123,87,
         123,91,122,95,120,97,120,98,124,104,
         119,98,113,98,113,98,110
300 DATA 97,110,97,107,95
310 DATA 4,71,31,73,30,73,30,74,28
320 DATA 4,82,68,85,72,85,72,87,71
```

```
330 DATA 6,83,101,84,104,86,105,88,105,89,
         102,89,100
340 DATA 3,118,81,118,83,120,85
350 DATA 21,75,95,74,100,74,107,76,112,76,
         112,80,115,80,115,83,116,83,116,87,
         116,87,116,92,113,95,109,95,109,98,
         104,99,100,99,100,99,96,99,96,98,92,
         98,92
360 DATA 2,86,100,86,101
```

Let in dit geval extra op regel 120. Tijdens het proberen kan er beter 120 GOTO 120 staan, waardoor wordt voorkomen dat Lorelei verloren gaat. Het prentje komt overigens zeer goed in aanmerking voor een plaats in de illustere rij van parels die we al geregen hebben. De keuze of het ook werkelijk deel gaat uitmaken van de verzameling, is aan u.
Terug naar de werkelijkheid. Hoe krijgen we de computer zó ver dat hij ons in staat stelt een beeldpunt op het beeldscherm te manipuleren? Elke MSX-computer is uitgerust met vier cursorbesturingstoetsen (pijltoetsen). De MSX-computer behandelt deze toetsen en een joystick op dezelfde manier. De uitleesopdracht voor de cursorbesturingstoetsen is:

STICK(0)

STICK(1) en STICK(2) kijken naar de stand van de joysticks. Als u de joysticks prefereert boven de cursorbesturingstoetsen, dan is daar natuurlijk geen enkel bezwaar tegen, maar dan moeten wel de in de volgende programma's voorkomende STICK-opdrachten van het goede nummer worden voorzien. De STICK(0)-opdracht kijkt dus welke pijltoetsen zijn ingedrukt. Op welke manier dat gaat, is in afbeelding 13-1 goed te zien.



Afbeelding 13-1 De richtingen plus bijbehorende cursor-toetsen.

Elke richting komt overeen met een nummer. Een kort programma illustreert het gebruik.

```
10  CLS:KEYOFF
20  A=STICK(0)
30 LOCATE l5,10:PRINT A
40 GOTO 20
```

Met wat extra regels kan er al een figuurtje over het scherm worden bewogen.

```
10 CLS:KEYOFF
20 A=STICK(0)
30 IF A=3 THEN X=X+1
40 LOCATE X,10
50 PRINT CHR$(249)
60 GOTO 20
```

Regel 30 kan voor alle richtingen worden herhaald. Het is echter aantrekkelijker om een en ander te combineren.

```
10 SCREEN l:CLS:KEYOFF
20 A=STICK(0)
30 IF A=2 OR A=3 OR A=4 THEN X=X+1
40 IF A=6 OR A=7 OR A=8 THEN X=X-1
50 IF A=8 OR A=l OR A=2 THEN Y=Y-1
60 IF A=4 OR A=5 OR A=6 THEN Y=Y+1
70 LOCATE X,Y
80 PRINT CHR$(249)
90 GOTO 20
```

De werking van de vuurknop missen we nog. Voeg daarom een regel 85 aan het programma toe:

```
IF  STRIG(0)   THEN  PRINT   CHR$(204)
```

De spatiebalk heeft hierbij de functie van vuurknop. Als de spatiebalk ingedrukt is, levert STRIG(O) het getal -1 op. In de binaire algebra heet dat negatieve logica. Als de uitdrukking waar is (de knop is ingedrukt), dan is de waarde van die uitdrukking -1.
Op deze manier kunnen we ook met getallen werken. Als A=3 dan is (A=3) = -1, of met woorden: Ais A=3 waar is, dan heeft de uitdrukking (A=3) de waarde -1.
Met deze kennis van negatieve logica is het vorige programma op elegante wijze te herschrijven.

```
10 SCREEN l:CLS:KEYOFF
20 A=STICK(0)
30 X=X+(A=6)+(A=7)+(A=8)-(A=2)-(A=3)-(A=4)   .
40 Y=Y+(A=2)+(A=1)+(A=8)-(A=4)-(A=5)-(A=6)
```

```
50  IF Y>22 THEN Y=22
60  IF Y<0 THEN Y=0
70  IF X<0 THEN X=0
80  IF X>28 THEN X=28
90  LOCATE X,Y
100 PRINT CHR$(249);
110 IF STRIG(0)=-1 THEN PRINT "VUUR!"
120 GOTO 20
```

We hebben nu voldoende kennis verzameld om het tekenprogramma te kunnen volgen. Dit relatief kleine programma is in staat om een gigantische hoeveelheid coördinaten van beeldpunten te genereren en op cassetteband op te slaan, zonder dat we daar verder zelf iets aan hoeven te doen.

```
10 REM vrij tekenen
20  STOP ON
30  DIM X(200):DIM Y(200)
40  SCREEN 2:COLOR 15,4,4:CLS
50  XN=0:YN=0:XO=0:YO=0
60  X=0:Y=0
70  PSET (XN,YN)
80  X=XN:Y=YN
90  A=STICK(0)
100 XN=X+(A=6)+(A=7)+(A=8)-(A=2)-(A=3)-(A=4)
110 YN=Y+(A=2)+(A=1)+(A=8)-(A=4)-(A=5)-(A=6)
120 PRESET(X,Y):PSET(XN,YN)
130 ON STOP GOSUB 220
140 IF STRIG(0)=-1 THEN 150 ELSE 80
150 X(I)=XN:Y(I)=YN
160 I=I+1
170 IF I=200 THEN 170
180 LINE(XO,YO)-(XN,YN)
190 IF X0=0 AND Y0=0 THEN LINE (XO,YO)-(XN,YN),4
200 XO=XN:YO=YN
210 GOTO 80
220 SCREEN O:CLS
230 Q=I
240 FOR I=0 TO Q-l
250  PRINT X(I);Y(I);
260 NEXT I
270 PRINT:PRINT"OPSLAAN ? (J/N)";
280 B$=INPUT$(1)
290 IF B$="J" OR B$="j" THEN 300 ELSE 350
300 OPEN"CAS:"FOR OUTPUT AS «l
310 PRINT#1,Q
320 FOR I=0 TO Q-l
330  PRINT#1,X(I),Y(I)
340 NEXT I
350 CLOSE
```

```
360 SCREEN 0:PRINT"Nieuwe lijn ? (J/N)";
370 A$=INPUT$(1)
380 IF A$="J" OR A$="j" THEN RUN
390 IF B$="J" OR B$="j" THEN 400 ELSE END
400 ST=300
410 OPEN"CAS:" FOR OUTPUT AS #1
420 PRINT#1,ST
430 CLOSE
440 END
```

Het programma is zoals gezegd eenvoudig in het gebruik. Behalve de computer, geladen met het programma, hebben we nog enkele andere attributen nodig. Om te beginnen de afbeelding die we op het scherm willen brengen. Het effect is het mooist als de voorstelling ongeveer dezelfde afmetingen heeft als het beeldscherm. We kunnen de tekening dan op ware grootte op het beeldscherm overnemen. De tekening trekken we met een viltstift over op een stuk helder huishoudfolie. Elk helder folie is goed. Het folie met de overgetrokken lijntekening wordt met een plakbandje op de buis geplakt. Dat gaat het beste als de schermrand een afwijkende kleur heeft.

```
SCREEN  2:COLOR,4,5:CLS
```

Nu kan de juiste plaats van de afbeelding beter worden bepaald. Het tekenprogramma kan nu worden gestart met RUN (F5). Het scherm wordt blauw en linksboven staat een witte punt. Deze punt kan met de pijltoetsen over het gehele scherm worden bewogen. We brengen de punt op de plaats waar een lijn begint en drukken dan op de spatiebalk. Vervolgens wordt de punt een stukje verderop op de lijn gezet en de spatiebalk wordt weer ingedrukt. Daar is het eerste lijnstukje. We werken zo de hele lijn af. Op rechte stukken verbinden we punten ver uit elkaar, op sterk gekromde lijnen leggen we de punten heel dicht bij elkaar. De 'tekenpunt' kan vrijelijk over het scherm worden bewogen tot u er zeker van bent dat hij op de juiste plaats staat. Druk dan pas op de spatiebalk. Is de lijn in zijn geheel op het scherm overgenomen, druk dan op:

```
CTRL/STOP
```

Alle 'aangeklikte' coördinaten komen in beeld en de vraag OPSLAAN ? (J/N) verschijnt. Als er een fout gemaakt is, typen we N en beginnen gewoon opnieuw. Is alles goed gegaan, dan zetten we de bandrecorder op opname (SAVE/REC) en beantwoorden de vraag met J. De getekende lijn wordt opgeslagen. Het programma vraagt vervolgens of we een nieuwe lijn willen toevoegen on start zichzelf opnieuw. We bewegen de tekenpunt naar het begin van een volgende lijn. Op deze wijze werken we alle lijnen afzonderlijk af. Deze methode heeft als voordeel dat een bepaald lijnstuk altijd op het bandje terug is te vinden en, indien nodig, ook achteraf nog kan worden veranderd. Na het laatste lijnstuk zet de computer een stopcode op de band (het getal 300). Begin niet gelijk het plafond van de Sixtijnse kapel te tekenen. Maak bijvoorbeeld een plattegrond van uw huis. Of gewoon een cirkel. Een lijn mag maximaal tweehonderd punten

bevatten. U zult ze maar zelden allemaal nodig hebben. Mocht het gebeuren dat u ze toch heeft verbruikt, dan merkt u dat de pijl-
besturing niet meer werkt. Gebruik CTRL/STOP en J om het afgewerkte lijn-deel op te slaan maar onthoud even waar u gebleven was. Het resterende deel wordt dan als een nieuwe lijn ingevoerd.
Nu de eerste tekening gemaakt is, willen we deze ook weer vanaf het bandje in de computer terug hebben. Daartoe dient het volgende, wel zeer eenvoudige, programma.

```
10 REM tekening van tape naar scherm
20 CLS
30 PRINT'Tape klaar... druk op een toets"
40 A$=INPUT$(1)
50 DIM X(200):DIM Y(200)
60 SCREEN 2:COLOR l,10,10:CLS
70 OPEN"CAS:" FOR INPUT AS #1
80 INPUT#!,Q
90 IF Q=300 THEN 170
100 FOR I=1 TO Q
110 INPUT#1,X(I),Y(I)
120 IF I<2 THEN 140
130 LINE(X(I-1),Y(I-1))-(X(I),Y(I))
140 NEXT I
150 CLOSE
160 GOTO 70
170 CLOSE
180 RUN"CAS:"              (180 GOTO  180)
```

Zorg er bij het testen voor dat de cassetterecorder aanstaat (op afspelen) en de band aan het begin van de opgenomen lijnstukken. Lijn voor lijn wordt de tekening op het beeldscherm opnieuw in elkaar gezet. Een prachtige show omdat de computer zelf de recorder start en stopt om het volgende lijnstukje op te halen.
Als we een tekening als pareltje aan de ketting willen rijgen, dan zetten we eerst het zojuist beschreven uitvoerprogramma op de band. Direct daar achteraan komt de tekening (de tape-uitvoer van hettekenprogramma). Het bandje moet dus een keer worden verwisseld om hettekenprogramma, dat op een andere band staat, te kunnen laden. Als het uitvoerprogramma alle informatie van de band heeft gehaald, treft het vanzelf de volgende parel aan, die door regel 180 wordt geladen en gestart. Voor elke opgeslagen tekening komt dus het uitvoerprogramma te staan.
De coördinatenparen kunnen ook op papier worden afgedrukt. We volstaan met een eenvoudig programma. De getallenrij kan dan, zoals in het Lorelei-programma, met de hand als data aan een programma worden toegevoegd. Iets meer werk, dat wel. Een handige programmeur zal er niet veel moeite mee hebben het volgende programma zodanig te wijzigen dat het programma zelf de DAT A-regels produceert. Als voorbeeld kan de sprite-ontwerper uit hoofdstuk 15 worden gebruikt. Dat programma genereert automatisch de benodigde DATA-regels.

```
10 REM co-ordinatenuitvoer naar printer
20  CLS:KEY OFF
30  PRINT'Tape klaar... druk op toets"
40  A$=INPUT$(1)
50  CLS
60  PRINT'Printer klaar... druk op toets"
70  A$=INPUT$(1)
80  CLS
90  DIM X(200):DIM Y(200)
100 LOCATE 9,10:PRINT "WACHT OP UITVOER"
110 OPEN"CAS:" FOR INPUT AS *1
120 INPUT#l,Q
130 IF Q=300 THEN CLOSE:END
140 LPRINT
150 FOR I=1 TO Q
160   INPUT#1,X(I),Y(I)
170   LPRINT X(I);Y(I);
180 NEXT I
190 CLOSE
200 LPRINT:LPRINT
210 GOTO 110
```

Op de printer verschijnen de X- en Y-coördinaten voor elk lijnstuk in paren achter elkaar. De rij wordt per lijnstuk voorafgegaan door een getal dat aangeeft hoeveel coördinatenparen er in het desbetreffende lijnstuk aanwezig zijn. Dit getal gebruikt de computer om de afzonderlijke lijnstukken te kunnen herkennen.
Als de stroom getallen ingelezen en afgedrukt is, kan het programma op de normale manier worden onderbroken met CTRL/STOP. Het kan gebeuren dat de rij laatste getallen pas wordt afgedrukt als het programma al is gestopt. Zet de printer daarom niet voortijdig uit.
Afgezien van Lorelei hebben we in dit hoofdstuk geen pareltjes aan de ketting toegevoegd. We hebben nu echter wel het gereedschap om zelf zeer individuele pareltjes te ontwerpen. Computertekeningen die achteraf weer kunnen worden opgeroepen, zouden wel eens erg bruikbaar kunnen blijken te zijn bij instructieprogramma's, bij het ontwerpen van achtergronden voor spelletjes of voor het maken van grote (afwijkende) letters. De tekeningen kunnen zonder meer achteraf verder worden bewerkt. Zo kunnen ze over het scherm worden verplaatst door bij de coördinaten een getal op te tellen of er een getal af te trekken. Ook is het achteraf inkleuren van de tekening mogelijk. Om dit laatste te demonstreren, geeft het volgende programma een letter weer waarvoor slechts weinig coördinaten nodig zijn. Dit om al te veel typewerk te voorkomen.

```
10 REM Assen
20 SCREEN 2:COLOR l,10,10:CLS
30 FOR I=1 TO 9
40 READ X(I),Y(I)
50 IF I<2 THEN 80
```

```
60    LINE(X(I-1),Y(I-1))-(X(I),Y(I)),5
70    LINE(X(I-1)+70,Y(I-1)+70)-(X(I)+
70,Y(I)+70),2 80  NEXT I
90  PAINT(30,1),5
100 PAINT(100,71),2
110 DATA 29,0,104,0,97,26,70,26,59,96,40,96,
         51,26,24,26,29,0
120 RUN"CAS:" (120 GOTO 120)
```

Eindelijk weer eens een programma dat aan de ketting kan. Het laat twee zaken zien. De DATA-regel bevat de coördinaten van een grote letter T. Regel 60 zorgt er dan ook voor dat op de opgegeven plaats een T op het scherm wordt gezet. Regel 70 echter verhoogt alle coördinaten met 70. Dezelfde letter T wordt dus 70 beeldpunten naar rechts en 70 beeldpunten naar beneden nogmaals afgebeeld. Bovendien gebeurt dit in een andere kleur. In de regels 90 en 100 krijgen de letters elk hun eigen kleurtje. Deze snelle letters kunnen nog sneller.

```
10  REM TT
20  SCREEN 2:COLOR 1,10,10:CLS
30  FOR 1=1 TO 9
40   READ X(I),Y(I)
50    IF I<2 THEN 70
60    LINE(X(I-1),Y(I-1))-(X(I),Y(I))
70  NEXT I
80  PAINT(30,1)
90  FOR 1=1 TO 9
100   IF I<2 THEN 140
110   FOR T=1 TO 10
120     LINE(X(I-1)+10*T,Y(I-1)+5*T)-
(X(I)+10*T,Y(I)+5*T),2 130   NEXT T
140 NEXT I
150 DATA 29,0,104,0,97,26,70,26,59,96,40,96,
         51,26,24,26,29,0
160 RUN"CAS:" (160 GOTO 160)
```

Ze worden nu in volle vaart op het scherm herhaald. Het teken- en het weergeefprogramma uit dit hoofdstuk bieden zoveel grafische mogelijkheden dat het de moeite waard is ze op een afzonderlijk bandje te bewaren. Op datzelfde bandje kan ook de sprite-ontwerper uit het volgende hoofdstuk worden gezet. Een duidelijke sticker op de cassette zorgt ervoor dat dit bandje met grafische hulpgereedschappen altijd gemakkelijk te herkennen en snel terug te vinden is.

14 Flitsende geesten

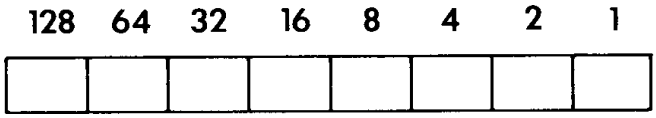
Sprites, het magisch toverwoord voor computerfanaten. Pacman, Zaxxon, Ghost Busters, noem maar op. Overal zien we ze, kleine figuurtjes die over het scherm bewegend grote vernielingen aanrichten, vitaminepillen eten of in afgronden storten. Sprites, moeilijk te begrijpen, moeilijk te maken en moeilijk te manipuleren. Alhoewel... niet in MSX-BASIC! Kijk zelf maar.

```
10 REM sprite
20  SCREEN 2,1:COLOR ,l,l:CLS
30  S$=""
40  FOR 1=1 TO 8
50   READ A$
60   A=VAL("&H"+A$)
70    S$=S$+CHR$(A)
80  NEXT I
90  DATA 18,3C,24,7E,DB,7E,24,00
100 SPRITE$(0)=S$
110 X=120:Y=85
120 FOR T=l TO 500
130   X=X+RND(l)*10-5
140  Y=Y+RND(l)*8-4
150   PUT SPRITE O,(X,Y),6,0
160 NEXT T
170 CIRCLE(X+7,Y+7),20,5
180 RUN"CAS:"     (180 GOTO 180)
```

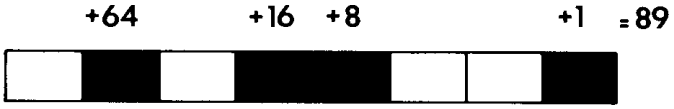
Waar komt die engerd nu weer vandaan? Van Mars of van Pluto, van een verre asteroïde? Nee hoor, gewoon uit regel 90. Hoe sprites worden opgebouwd is in veel verschillende boeken en MSX-handleidingen terug te vinden. Afhankelijk van de voorkeur voor getallenstelsels kunnen we sprites op drie manieren definiëren. In decimale getallen, in hexadecimale getallen (zoals in het voorgaande programma) en in binaire getallen.
Iedereen kan zo zijn of haar eigen voorkeur hebben. We zullen hier de binaire en de decimale schrijfwijze even kort herhalen aan de hand van de griezel uit het programma. Er zijn twee soorten sprites. Het verschil zit 'm uitsluitend in de afmeting. Er zijn sprites van 8 x 8 beeldpunten en er zijn sprites van 16x16 beeldpunten. De onderhavige geest is een 8 x Sfiguur. Binair is een sprite heel eenvoudig te beschrijven. Als we uitgaan van onze geest van 8x8 beeldpunten dan ziet het basisontwerp er als volgt uit:

```
---XX--- DATA 00011000 ( 24)
--XXXX-- DATA 00111100 ( 60)
--X--X-- DATA 00100100 ( 36)
-XXXXXX- DATA 01111110 (126)
XX-XX-XX DATA 11011011 (219)
-XXXXXX- DATA 01111110 (126)
--X--X-- DATA 00100100 ( 36)
________ DATA 00000000 (  0)
```

De achter het figuurtje afgedrukte regels zijn de DATA-regels zoals die in het programma worden opgenomen als de binaire vorm wordt gebruikt. De getallen tussen haakjes geven de overeenkomstige decimale waarde aan. Die is eenvoudig af te leiden:

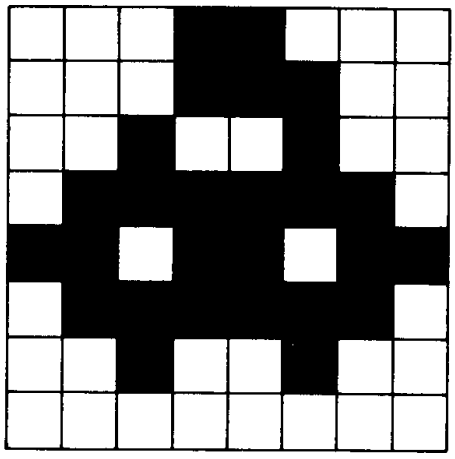


Voor elk vakje dat zwart wordt gemaakt, tellen we het erboven staande getal op. Voorbeeld:



Het zal duidelijk zijn dat de decimale ontwerpmethode het minste typewerk vraagt, maar dat de binaire methode een herkenbaar beeld oplevert. Tenminste, als de DATA-regels onder elkaar worden geplaatst. Vergelijk:

```
100 DATA 00011000
110 DATA 00011100
120 DATA 00100100
130 DATA 01111110
140 DATA 11011011
150 DATA 01111110
160 DATA 00100100
170 DATA 00000000
```

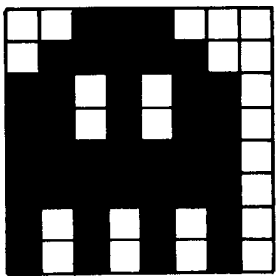


met:

```
100  DATA 24,28,36,126,219,126,36,0
```

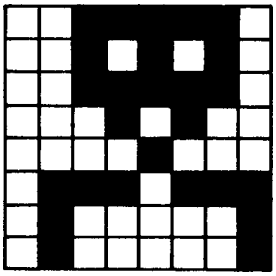
Vanwege de toch wel erg compacte schrijfwijze zullen we ons verder beperken tot de decimale sprite-notatie.
Een sprite laat zich het eenvoudigst ontwerpen op een vel ruitjespapier. Dat kan ook zelf getekend worden. Maak de hokjes die te zamen een SPRITE moeten vormen zwart en bekijk het resultaat vanaf enige afstand. Dan krijgt u een goed beeld van hoe de sprite er op het scherm uit zal gaan zien. Het is ook handig om vierkantjes uit zwart karton te knippen, zodat er gemakkelijk wat mee kan worden geschoven. Een nog betere methode om sprites te ontwerpen en visueel te beoordelen, wordt verderop in dit hoofdstuk beschreven. Eerst zullen we een paar sprites maken en op het scherm toveren. Het meest dankbaar zijn de spookachtige figuurtjes, maar ook vliegtuigen en race-auto's zijn geliefd.

Een geest om bang van te worden:



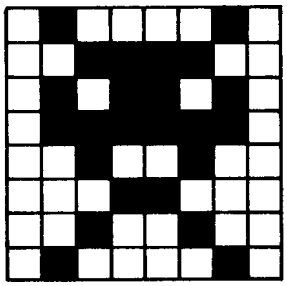
```
DATA 56,124,214,214,254,254,170,170
```

Zijn compagnon:



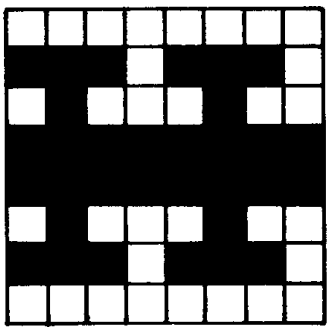
```
DATA 62,42,62,20,8,119,65,65
```

En hun baas:



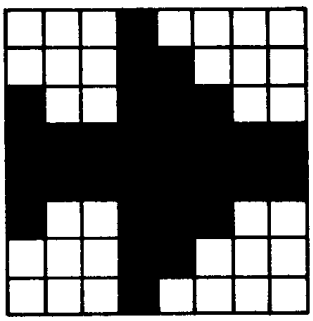
```
DATA 66,60,90,126,36,24,36,66
```

Een race-auto:



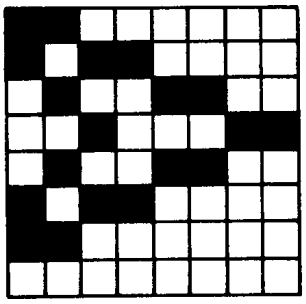
```
DATA  0,238,68,255,255,68,238,0
```

Een vliegtuig:



```
DATA   16,24,156,255,255,156,24,16
```

En tot slot een raket:



```
DATA  192,176,76,35,76,176,192,1
```

Dit zijn allemaal heel eenvoudige sprites. Toch kunnen we er al aardige dingen mee doen. Voor we zover zijn, bekijken we hoe een sprite in een programma wordt gebruikt. Elke sprite krijgt een eigen naam en wordt opgebouwd uit een string. In die string komen de DATA-codes te staan. Omdat die codes uit getallen bestaan, moeten we die eerst omzetten in een string. Voor de raket-sprite ziet het geheel er dan zó uit:

```
SPRITE$(1)=CHR$(192)+CHR$(176)+CHR$(76)+CHR$(35)+
CHR$(76)+CHR$(176)+CHR$(192)+CHR$(0)
```

Dat ziet er niet als erg aantrekkelijk typewerk uit. Het is beter de computer zelf voor zijn zaakjes te laten zorgen. Een programma dat zelf de DATA-regel leest, er een sprite van maakt en die op het scherm zet, heeft voor decimale DATA de volgende vorm:

```
10 REM geest
20 SCREEN 2,1:COLOR,1,1:CLS
30 FOR I=1 TO 8
40 READ S
50 S$=S$+CHR$(S)
60 NEXT I
70 SPRITE$(1)=S$
80 PUT SPRITE 0,(120,100),10,1
90 GOTO 90
100 DATA 56,124,214,214,254,254,170,170
```

Enkele opmerkingen kunnen verhelderend zijn. Het nummer van de sprite mag van 0 tot 256 lopen. Let wel, dat geldt voor de kleintjes waar we momenteel mee bezig zijn. Verderop komt het echte werk met 16x16 sprites, waarvan er 'slechts' 64 verschillende voor mogen komen.
In de PUT SPRITE-opdracht staat het sprite-nummer helemaal achteraan. Het eerste nummer heeft betrekking op het vlak waarin de sprite acteert. Er zijn 31 vlakken achter elkaar mogelijk. Een sprite in vlak 1 gaat op het scherm ook echt achter een sprite in vlak 0 langs. Een sprite in het laatste vlak (31) schuift achter alle andere sprites langs.
Een paar spookjes zoals die in dit hoofdstuk beschreven zijn, treden op in het geestenprogramma.

```
10 REM spokendans
20  SCREEN 2,0:COLOR ,4,4:CLS
30  FOR I=1 TO 8:READ S:S$=S$+CHR$(S):NEXT I
40  FOR I=1 TO 8:READ G:G$=G$+CHR$(G):NEXT I
50  SPRITE$(0)=S$
60  SPRITE$(1)=G$
70  X=0:Y=0
80  FOR I=1 TO 15 STEP .2
90   X=X+RND(1)*10-RND(1)*5
100  Y=Y+RND(1)*5-RND(1)*2
110   X1=30*COS(I):X2=2*X1
120   Y1=30*SIN(I):Y2=2*Y1
130   PUT SPRITE 2,(X,Y),11,0
140   PUT SPRITE 1,(X+X1,Y+Y1),9,1
150   PUT SPRITE 0,(X+X2,Y+Y2),14,1
160 NEXT I
170 DATA 56,124,214,214,254,254,170,170
180 DATA 62,42,62,20,8,119,65,65
190 RUN"CAS:" (190 GOTO 190)
```

De inmiddels overbekende laatste regel rijgt de spoken aan de rode draad die door dit boek loopt.
De behandelde sprites zijn weliswaar leuk, maar toch verre van spectaculair. We kunnen er iets aan doen door ze wat op te blazen. Het enige wat we daarvoor aan het programma hoeven te wijzigen, is de SCREEN-opdracht in regel 20. Probeer maar eens:

```
20  SCREEN   2,1:COLOR,4,4:CLS
```

Het formaat scheelt aanzienlijk.
De SCREEN-opdracht kent als tweede variabele een getal dat bepaalt welke sprites worden geaccepteerd (8 x Sof 16 x 16) en de afmeting waarop ze afgebeeld dienen te worden.

SCREEN 2,0	8 x 8 normaal
SCREEN 2,1	8 x 8 tweemaal vergroot
SCREEN 2, 2	16 x 16 normaal
SCREEN 2,3	16 x 16 tweemaal vergroot

Jammer dat de sprites met verschillende afmetingen niet door elkaar te gebruiken zijn. Om de allermooiste sprites te kunnen maken, is het 16x16-formaat ideaal. Het ontwikkelen van sprites met deze afmeting is echter heel wat lastiger dan bij hun kleinere broertjes het geval is. Ook hier komt de computer te hulp. In het volgende hoofdstuk wordt een geraffineerd ontwerpprogramma besproken dat ons in staat stelt voor elke mogelijke vorm de best lijkende sprite te ontwerpen. Het lijkt wel wat op het tekenprogramma uit hoofdstuk 13.

15 Sprite-ontwerper

In tegenstelling tot de meeste programma's uit dit deel van het boek is het sprite-ontwerpprogramma tamelijk uitgebreid. Het presteert dan ook een en ander. Het tekent een patroon van zestien bij zestien velden. Hierin wordt met de pijlbesturing het te vullen vlak aangewezen en met een druk op de spatiebalk wordt het aangewezen vlak zwart gekleurd. Indien we een verkeerd vlak hebben gevuld, kunnen we dit vlak weer 'leeg' maken door het nogmaals met de spatiebalk aan te klikken. Als de sprite compleet is, berekent het programma de decimale getallen die de sprite vastleggen. Bovendien genereert het programma zelfstandig twee DATA-regels waarin de getallen staan. Deze kunnen ook op papier worden weergegeven. De DATA-regels die op het scherm worden afgebeeld, zijn al van regelnummers voorzien. Deze kunnen als afzonderlijke programma-onderdelen worden opgeslagen op een cassettebandje. Zo kunt u een aardige sprite-collectie opbouwen. We kunnen er ook andere programmaregels aan toevoegen die de zojuist ontworpen sprite een eigen leven laten leiden (lijden).
Voordat u dit programma test, moet u het wel 'saven'. Aan het einde van het programma wordt het programma gewist, opdat alleen de DATA-regels met de sprite-gegevens in het geheugen van de computer blijven staan.

```
10  REM ** 16x16 Sprite-ontwerper ***
20  KEY OFF:SCREEN O:COLOR15,4,4:CLS
30  PRINT"Sprite-afmeting"
40  LOCATE 10,10:PRINT"l Klein (Scherm 2)"
50  LOCATE 10,12:PRINT"2 Groot (Scherm 3)"
60  A$=INPUT$(1):A=VAL(A$)
70  IF A<1 OR A>2 THEN 20
80  A=A+1
90  SCREEN 2,A:COLOR l,4,4:CLS
100 DIM V(2,31):DIM C(8,31)
105 REM **Bij einde sprite-ontwerp: 600
110 STOP ON
120 ON STOP GOSUB 600
125 REM **0ntwerpstramien-omlijsting
130 OPEN"GRP:" FOR OUTPUT AS #1
140 PSET(40,0),4
150 COLOR 15
160 PRINT#1,"SPRITE ONTWERP"
170 COLOR l
175 REM **0ntwerpstramien-raster
180 FOR 1=1 TO 17
190   LINE(32,32+I*8)-(160,32+I*8)
200   LINE(24+I*8,40)-(24+I*8,168)
210   LINE(96,32)-(96,176)
220 NEXT I
```

```
230 PSET(166,0),1:LINE -STEP(75,50),l,BF
235 REM **Cursor-besturing met pijltoetsen
240 XN=36:YN=44
250 PSET(XN.YN)
260 X=XN:Y=YN
270 A=STICK(0)
280 XN=X+8*(A=6)+8*(A=7)+8*(A=8)-8*(A=2)-
8*(A=3)-8*(A=4)
290 YN=Y+8*(A=2)+8*(A=1)+8*(A=8)-8*(A=4)-
8*(A=5)-8*(A=6)
295 REM **Rand van het ontwerpraster
300 IF XN<36 THEN XN=36
310 IF XN>156 THEN XN=156
320 IF YN<44 THEN YN=44
330 IF YN>164 THEN YN=164
335 REM **Cursor op de goede plaats zetten
340 PRESET(X,Y):PSET(XN,YN) 350 PRESET(X-l,Y-1):PSET(XN-1,YN-1)
360 PRESET(X+1,Y+1):PSET(XN+1,YN+1)
370 PRESET(X+l,Y-1):PSET(XN+1,YN-1)
380 PRESET(X-1,Y+1):PSET(XN-1,YN+1)
390 IF STRIG(O) THEN GOSUB 410
392 REM **Spatiebalk ingedrukt?
394 REM   Ja: subroutine vul hokje
396 REM  Nee: pijltoetsen uitlezen
400 GOTO 260
405 REM **Subroutine vul hokje
410 PAINT(XN+1,YN)
420 X=(XN-36)/8:Y=(YN-44)/8
430 IF X>=8 THEN A=2:Xl=X-8:Y1=Y+16
ELSE A=1:X1=X:Y1=Y
432 REM **Plaatsbepaling gevuld hokje
434 REM  en bepalen of hokje al gevuld
436 REM  is. Zo ja, hokje weer
438 REM   leegmaken(530)
440 IF C(X1,Y1)<>0 THEN GOSUB 530
ELSE 570
445 REM **String-uitdrukking voor sprite 450 S$=""
460 FOR I=1 TO 2
470   FOR J=0 TO 15
480    S$=S$+CHR$(V(I,J))
490 NEXT J,I
500 SPRITE$(0)=S$
504 REM **Sprite op het voorbeeldscherm
506 REM   zetten
510 PUT SPRITE O,(185,5),10,O
515 REM **Terug naar cursorbesturing
520 RETURN
524 REM **Gevuld hokje weer leegmaken
526 REM   en "sprite-getalen" aanpassen
530 C(X1,Y1)=0
540 V(A,Y)=V(A,Y)-2~(7-Xl)
550 LINE(XN-4,YN-4)-(XN+4,YN+4),4,BF
```

```
555 LINE(XN-4,YN-4)-(XN+4,YN+4),l,B
560 RETURN
565 REM **"Sprite-getallen" aanpassen
570 C(X1,Y1)=2^(7-X1)
580 V(A,Y)=V(A,Y)+2^(7-Xl)
590 RETURN
600 REM **Uitvoer
605 REM **Presentatie sprite-getallen
610 PSET(166,0),1:LINE -STEP(75,180),l,BF
620 FOR I=0 TO 15
630   PSET(170,44+8*I),4:COLOR 10:
PRINT #1,V(1,I) 640   PSET(210,44+8*I),4:COLOR 10:
PRINT #1,V(2,I);
650   COLOR l
660 NEXT I
670 PSET(0,0),4:COLOR 10:PRINT#1,
STRING$(22,219)
680 PSET(0,0),4:COLOR 1:PRINT#1,
"Printer uitvoer (J/N)"
690 A$=INPUT$(1)
700 IF A$="J" OR A$="j" THEN 710 ELSE 780
705 REM **Uitvoer sprite-getallen op papier
710 LPRINT"DATA";
720 FOR I=0 TO 14:LPRINT V(1,I);",";:NEXT I
730 LPRINT V(1,15)
740 LPRINT"DATA";
740 FOR I=0 TO 14:LPRINT V(2,I);",";:NEXT I
760 LPRINT V(2,15)
770 CLOSE
775 REM **Uitvoer sprite-getallen op scherm
780 SCREEN O:COLOR 15,4,4:CLS
790 PRINT"8000 DATA";
800 FOR I=0 TO 14
802  PRINT STR$(V(1,I))+",";
804 NEXT I
810 PRINT STR$(V(1,15)):PRINT
820 PRINT"8010 DATA";
830 FOR I=0 TO 14 832  PRINT STR$(V(2,I))+",";
834 NEXT I
840 PRINT STR$(V(2,15)):PRINT
850 LOCATE 0,10
860 PRINT'Plaats cursor op regelnummers"
870 PRINT"en druk op <RETURN>"
880 PRINT
882 REM **Verwijdering programmaregels om
884 REM   in het geheugen alleen de sprite-
886 REM  getallen over te houden
890 PRINT'Plaats cursor op:"
900 PRINT"DELETE 10-870"
910 PRINT"___________"
920 PRINT"en druk op <RETURN>"
930 PRINT
```

```
940 PRINT"Zet op cassette met:"
950 PRINT"SAVE"
960  REM **Einde  programma
```

Door de cursor op de afgebeelde regel:

```
DELETE 10-870
```

te zetten, wordt het ontwerpprogramma verwijderd en resteren alleen nog de DATA-regels.
De sprite-ontwerper geeft de DATA-regels hoge nummers. Deze regels kunnen op cassette worden gezet met dezelfde opdracht waarmee alle in dit boek voorkomende programma's worden opgenomen:

```
SAVE"sprite"
```

Ze worden dan in de onverkorte ASCII-code weggeschreven en kunnen met MERGE"sprite" aan elk willekeurig programma worden toegevoegd. De uitvoer van het programma kan er bijvoorbeeld zó uitzien:

```
8000 DATA 1,2,4,8,1618,21,23,21,18,8,4,
          4,2,2,2 8010 DATA 192,32,16,8,4,36,84,116,84,36,
          136,144,16,160,16,0,160
```

Hieronder staat het programma om deze geheimzinnige geest zichtbaar te maken.

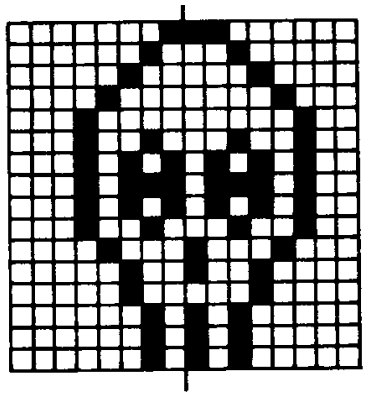
```
10 REM kleine 16x16 sprite
20 SCREEN 2,2:COLOR ,1,1:CLS
30 FOR I=1 TO 32:READ S:S$=S$+CHR$(S):NEXT I
40 SPRITE$(0)=S$
50 PUT SPRITE O,(120,80),10,O
60 RUN"CAS:" (60 GOTO 60)
8000 DATA 1,2,4,8,16,18,21,23,21,
18,8,4,4,2,2,2 8010 DATA 192,32,16,8,4,36,84,116,84,
36,136,144,16,160,160,160
```

Voor de vergrote versie hoeft slechts één cijfertje in de SCREEN-opdracht te worden veranderd.

```
20   SCREEN  2,3:COLOR   ,1,1:CLS
```

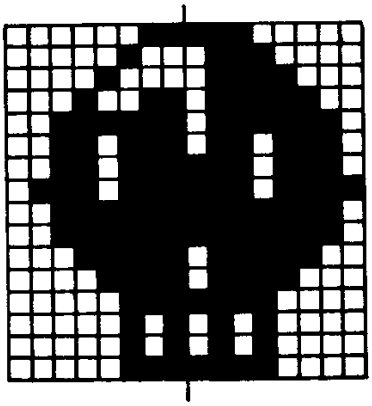
Ook kan die verandering in een programma plaatsvinden.

```
10  REM kleine en grote 16x16 sprite
20  FOR I=1 TO 32:READ S: S$=S$+CHR$(S) :NEXT I
30  FOR J= l TO 5
40    FOR SC=2 TO 3
50      SCREEN 2,SC:COLOR ,1,1:CLS
60     SPRITE$(0)=S$
70     PUT SPRITE O,(120,80),10,O
80     FOR Q=l TO 300:NEXT Q
90  NEXT SC,J
100 RUN"CAS:"         (100 GOTO 100)
8000 DATA 1,2,4,8,16,18,21,23,21,18,
          8,4,4,2,2,2
8010 DATA 192,32,16,8,4,36,84,116,84,
          36,136,144,16,160,160,160
```

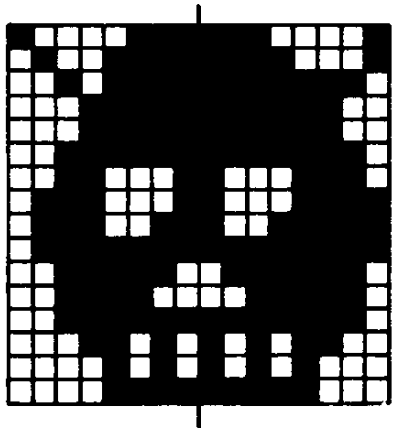


Drie andere sprites worden afgebeeld met de bijbehorende DATA-regels. Door de DATA-regels in het voorgaande programma te vervangen, doemen ze op uit de duisternis.

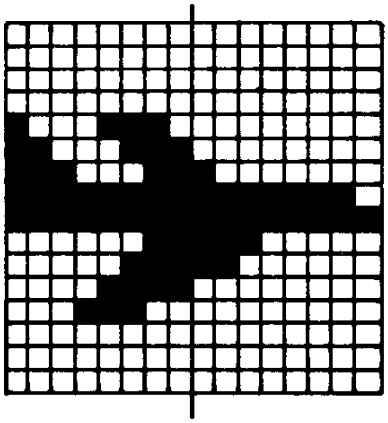
```
8000  DATA  1,2,4,9,31,31,29,29,61,31,31,
            15,7,2,2,3
8010  DATA  192,96,112,120,124,124,92,220,
            222,252,124,120,240,160,160,224
```



```
8000 DATA 135,79,63,31,63,56,120,121,127,
          127,62,63,63,26,10,15
8010 DATA 241,249,254,252,254,142,143,159,
          255,127,62,254,254,172,168,248
```



```
8000 DATA 0,0,0,0,135,195,225,255,255,3,7,
          15,28,0,0,0
8010 DATA 0,0,0,0,0,128,192,254,255,224,
          192,0,0,0,0,0
```

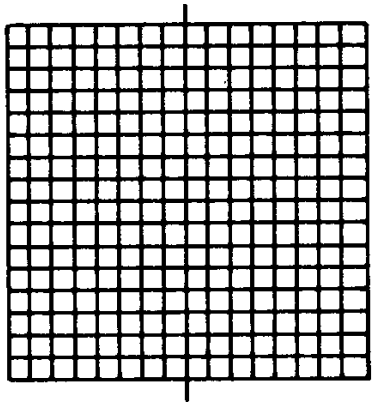


16 Spokendans en meer van dat moois

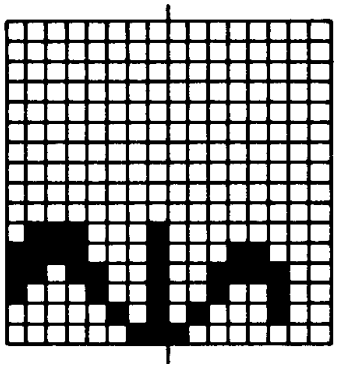
Sprites kunnen zich op 31 doorzichtige vlakken bevinden die achter elkaar staan. Ze kunnen dan ook achter elkaar langs bewegen.

```
10  REM spokendans
20   SCREEN 2,3:COLOR ,1,1:CLS
30   FOR I=1 TO 32:READ S:S$=S$+CHR$(S):NEXT I
40   SPRITE$(0)=S$:SPRITE$(1)=S$:SPRITES(2)=S$
50   FOR I=1 TO 515
60    PUT SPRITE 0,(I,70),10,0
70    PUT SPRITE 1,(256-1,80),7,1
80    PUT SPRITE 2,(2*I,90),9,2
90  NEXT I
100  RUN"CAS:"                (100  GOTO 100)
8000 DATA 1,2,4,9,31,31,29,29,61,31,31,15,7,2,2,3
8010 DATA 192,96,112,120,124,124,92,220,222,252
8020 DATA 124,120,240,160,160,224
```

Heel fraai demonstreren de demonen hoe ze zich achter elkaar kunnen verschuilen. Er is nog iets bijzonders op het scherm te zien. Een geest verdwijnt helemaal van het scherm, voordat hij aan de andere kant weer tevoorschijn komt. Het MSX-scherm heeft aan de randen een uitloopje, vergelijkbaar met de coulissen van een toneelpodium.
Bewegen de geesten wat beverig? Laat dan regel 90 eens weg. Supersnelle geesten flitsen dan over het scherm.
Doordat sprites zich op 31 schermen kunnen bevinden, kunnen we ook meer-kleuren-sprites maken. We zetten er dan een aantal op dezelfde plaats. Er is slechts één beperking: nooit meer dan vier sprites op een regel. De sprites met de hoogste schermnummers worden aangetast of weggelaten als er meer dan vier zijn.
Sprites hoeven echt niet altijd uit een duistere wereld te komen. Ook in een tuintje doen ze het uitstekend. Als voorbeeld van een meerkleuren-sprite zullen we een bloemetje ontwerpen. Omdat een meerkleuren-portretje van een bloem uit meerdere delen is opgebouwd, gebruiken we een sprite-ontwerp-raster.

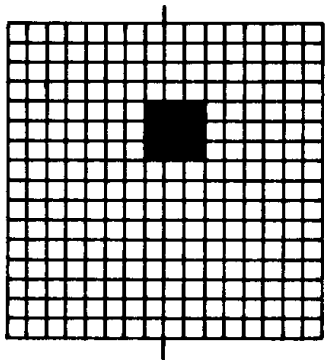


Als we een stapeltje fotokopieën van zo'n raster van 16 bij 16 hokjes maken, hoeven we slechts éénmaal een tekening te maken. Met kleurpotlood kan elk ingewikkeld ontwerp op redelijk eenvoudige wijze in elkaar worden gezet. Bij het maken van de sprite met behulp van het ontwerpprogramma is zo'n voorontwerp op papier een prachtig hulpmiddel. We beginnen met de groene bladeren van de bloem.



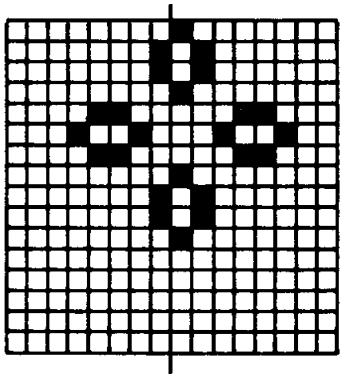
```
DATA  0,0,0,0,0,0,0,0,0,0,113,241,217,137,5,3
DATA  0,0,0,0,0,0,0,0,0,0,0,24,60,36,68,128
```

Dan is het hartje van de bloem aan de beurt:



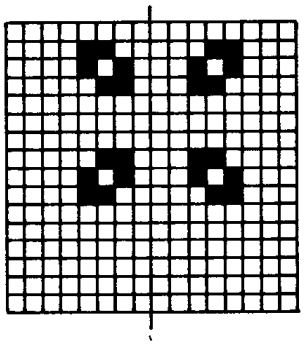
```
DATA  0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0
DATA  0,0,0,0,192,192,192,0,0,0,0,0,0,0,0,0
```

De horizontale en verticale bloemblaadjes zien er zó uit:



```
DATA  0,1,1,0,12,18,12,0,1,1,0,0,0,0,0,0
DATA  128,64,64,128,24,36,24,128,64,64,128,0,0,0,0,0
```

De diagonale bloemblaadjes completeren het geheel:

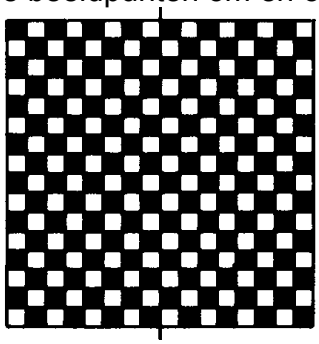


```
DATA  0,12,10,6,0,0,0,6,10,12,0,0,0,0,0,0
DATA  0,24,40,48,0,0,0,48,40,24,0,0,0,0,0,0
```

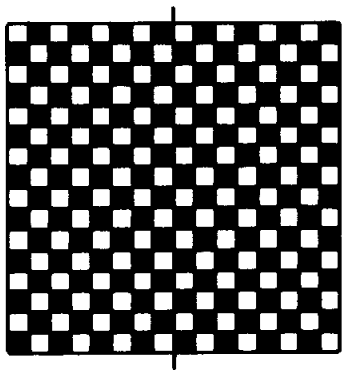
Het zojuist ontworpen bloemetje wordt in een pareltje gemonteerd:

```
10  REM bloem
20   CLEAR 300
30   SCREEN 2,3:COLOR ,2,10:CLS
40   CIRCLE(85,70),40,3
50   PAINT(85,70),3
60  FOR J=0 TO 3
70    FOR I=1 TO 32
80       READ S:S$=S$+CHR$(S)
90     NEXT I
100   SPRITE$(J)=S$
110   S$=""
120  NEXT J
130  FOR I=1 TO 60
140   PUT SPRITE 3,(I*1.3,I),12,0
150   PUT SPRITE 2,(I*1.3,I),9,1
160   PUT SPRITE 1,(I*1.3,I),4,2
170   PUT SPRITE 0,(I*1.3,I),5,3
180    FOR Q=1 TO 100:NEXT Q
190  NEXT I
200  RUN"CAS:"       (200 GOTO 200)
8000 DATA 0,0,0,0,0,0,0,0,0,0,113,241,217,137,5,3
8010 DATA 0,0,0,0,0,0,0,0,0,0,0,24,60,36,68,128
8020 DATA 0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0
8030 DATA 0,0,0,0,192,192,192,0,0,0,0,0,0,0,0,0
8040 DATA 0,1,1,0,12,18,12,0,1,1,0,0,0,0,0,0
8050 DATA 128,64,64,128,24,36,24,128,64,64,128,0,0
8060 DATA 0,0,0,0,12,10,6,0,0,0,6,10,12,0,0,0,0,0
8070 DATA 0,0,24,40,48,0,0,0,48,40,24,0,0,0,0,0,0
```

Dat kleurencombinaties het niet altijd goed doen, laat de bloem zien die onderweg is naar het perkje. Eenmaal in het perkje aangekomen, ziet het er allemaal een stuk fleuriger uit. Lijkt het niet erg veel op een kunstig borduurwerkje? (Misschien is het niet eens een gek idee om met de MSX-computer borduur-werkjes te ontwerpen.)
Bij het weglaten van regel 20 verschijnt er een foutmelding op het scherm. Normaal gesproken reserveert het MSX-systeem een aantal geheugenplaat-sen voor strings. Als die hoeveelheid niet voldoende is, kunnen we met CLEAR het aantal plaatsen opgeven dat wél toereikend is; in dit geval 300. Hoewel dit getal aan de hand van het programma uit te rekenen is, blijkt het vaak praktischer het gewoon te proberen. Verhoog het getal, beginnend bij tweehonderd (de normale toestand) steeds met 50 tot de foutmelding niet meer verschijnt. Als we het bloemetje meermalen willen afbeelden, is het zaak ervoor te waken dat er geen twee bloemen op één regel komen. Een bloem bestaat immers uit viersprites, het maximum vooreen regel. Schuin onder elkaar kan dus wel. Nu we toch met meerkleuren-sprites bezig zijn, is het aardig om het eerder besproken verf mengproces eens sprite-matig te bekijken. Stel we maken twee rasters, waarbij de beeldpunten om en om zijn ingevuld:



```
DATA  170,85,170,85,170,85,170,85,170,85,170,85,17,
      85,170,85
DATA  170,85,170,85,170,85,170,85,170,85,170,85,17,
      85,170,85
```

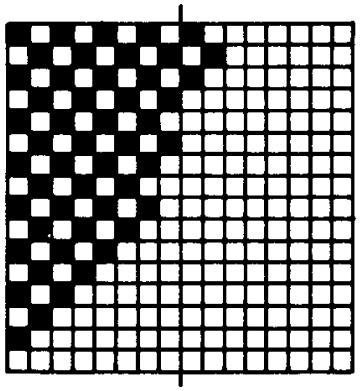


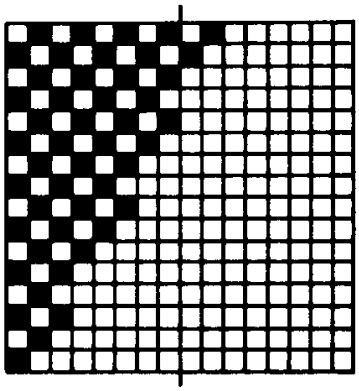
```
DATA  85,170,85,170,85,170,85,170,85,170,85,170,85,
      170,85,170
DATA  85,170,85,170,85,170,85,170,85,170,85,170,85,
      170,85,170
```

In een programma krijgen de rasters een eigen kleur en worden ze over elkaar heen geprojecteerd.

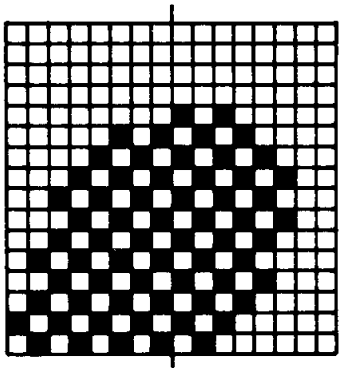
```
10  REM oranje sprite
20  SCREEN 2,2:COLOR,1,1:CLS
30  FOR J=0 TO l
40   FOR I=1 TO 32
50      READ S:S$=S$+CHR$(S)
60   NEXT I
70    SPRITE$(J)=S$
80    S$=""
90  NEXT J
100 PUT SPRITE 0,(80,80),10,O
110 PUT SPRITE 1,(80,80),6,1
120 RUN"CAS:" (120 GOTO 120)
8000 DATA 170,85,170,85,170,85,170,85,170,
          85,170,85,170,85,170,85
8010 DATA 170,85,170,85,170,85,170,85,170,
          85,170,85,170,85,170,85
8020 DATA 85,170,85,170,85,170,85,170,85,
          170,85,170,85,170,85,170
8030 DATA 85,170,85,170,85,170,85,170,85,
          170,85,170,85,170,85,170
```

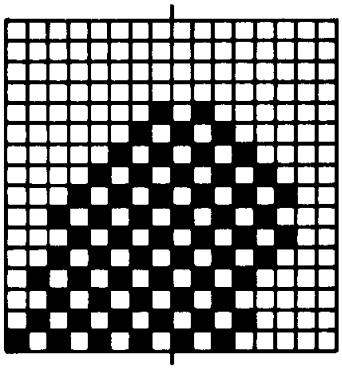
Een idee om eens te gebruiken, bijvoorbeeld in een spel met konijnen waarbij winterpeentjes het lokaas zijn. De kleur is er goed voor. Aan de afmeting ontbreekt echter iets.
Een hoogstandje van sprite-kunst zou een grote winterpeen zijn, in de kleur oranje, met nog een restje blad eraan. We gaan eerst weer met de rasters aan de slag. Omdat we lichtrood en donkergeel willen mengen, zetten we de beeldpunten om en om aan. Dat geldt voor de onderkant van de wortel:



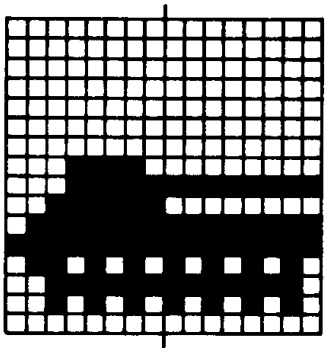


maar ook voor de bovenkant:





Voor het blad maken we de volgende sprite:



Een programma waarin de reuzenwortel een rol zonder happy end speelt, zou er uit kunnen zien als het volgende programma. Een vijfvoudige sprite in twee (eigenlijk drie) kleuren, dat is toch een mooi resultaat.

```
10 REM peen
20 SCREEN 2,3:COLOR 1,7,7:CLS
30 FOR J=0 TO 4
40 FOR I=1 TO 32
50 READ S:S$=S$+CHR$(S)
60 NEXT I
70 SPRITE$(J)=S$
80 S$=""
90 NEXT J
100 CIRCLE(60,110),30,11
110 PAINT(60,110),11
120 CIRCLE(45,70),30,11,,,6
130 PAINT(45,70),11
140 CIRCLE(62,70),30,11,,,6
150 PAINT(62,70),11
160 CIRCLE(61,100),8,1,,,2
170 PAINT(61,100),1
180 PSET(67,102),15
190 CIRCLE(-10,192),80,11
200 PAINT(1,191),11
```

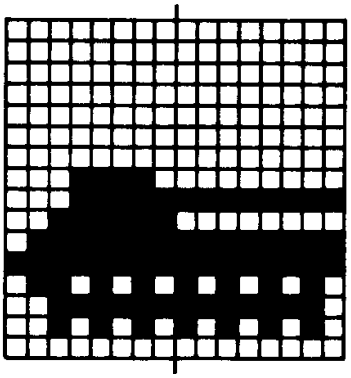
```
210 CIRCLE(80,110),11,1,4,5
220 FOR I=1 TO 100
222 X=RND(1)*180:Y=192-RND(l)*50:R=RND(1)*20
230 CIRCLE(X,Y),R,2
235 PAINT (X,Y),2
240 NEXT I
250 FOR I=1 TO 120
260 PUT SPRITE 4,(200,I+32),9,O
270 PUT SPRITE 3,(200,I+32),10,l
280 PUT SPRITE 2,(200,I),9,2
290 PUT SPRITE 1,(200,I),10,3
300 PUT SPRITE 0,(200,I),12,4
310 NEXT I
320 FOR Q=l TO 100:NEXT Q
330 PUT SPRITE 4,(200,208),9,O
340 PUT SPRITE 3,(200,208),10,l
350 PUT SPRITE 2,(200,208),9,2
360 PUT SPRITE 1,(200,208),10,3
370 PUT SPRITE 0,(58,115),12,4
380 RUN"CAS:"          (GOTO 380)
8000 DATA 170,85,170,85,170,84,170,84,168,80,
168,80,160,64,128,128
8010 DATA 128,0,128,0,0,0,0,0,0,0,0,0,0,0,0,0
8020 DATA 85,171,85,170,85,170,84,168,84,168,
         80,160,64,160,64,128
8030  DATA  64,128,0,0,0,0,0,0,0,0,0,0,0,0,0,0
8040  DATA 0,0,0,0,0,5,10,21,42,21,42,85,42,85,
          170,85
8050  DATA 0,0,0,0,160,80,168,84,168,84,168,80,
          160,80,160,64
8060  DATA 0,0,0,0,1,2,5,10,21,42,21,42,85,
          42,85,170
8070  DATA 0,0,0,0,64,160,80,168,84,168,80,168,
          80,160,80,160
8080  DATA  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
8090  DATA  241,51,63,29,25,0,0,0,0,0,0,0,0,0,0,0
```

Zo'n uitgebreide DATA-lijst is niet echt leuk om in te typen. Het kleinste foutje is vaak al fataal. De DATA-lijsten kunnen ook door de sprite-ontwerper worden gemaakt. Bij meervoudige sprites veranderen we de door het ontwerpprogramma opgegeven regelnummers (8000 en 8010) in opeenvolgende nummers (8000-8010,8020-8030 enz.). De regelparen 'saven' we als afzonderlijke programma's op een bandje. Achteraf kan er dan een programma van worden gemaakt met de MERGE-opdracht. Hiervoor is het nodig dat we niet de CSAVE- maar de SAVE-opdracht gebruiken. Daar zult u weinig moeite mee hebben, omdat we al onze pareltjes al met de SAVE-opdracht aan de ketting hebben geregen.
Er is nóg een aspect niet aan de orde geweest. In het beeldverhaal uit het vorige programma verdwijnt er plotseling een stuk wortel. Het zal iedereen duidelijk zijn wie daarvoor verantwoordelijk is. De eet-opdracht is te vinden in de regels 330 en 340. Als de Y-coördinaat van een sprite de waarde 208 of 209 heeft, verdwijnt

hij van het scherm. Let hierbij op! Als de Y-coördinaat 208 is, verdwijnen ook alle sprites met een hoger schermnummer. Dat is niet alleen handig als verdwijntruc, een sprite kan er ook onheilspellend mee aan het knipperen worden gebracht.

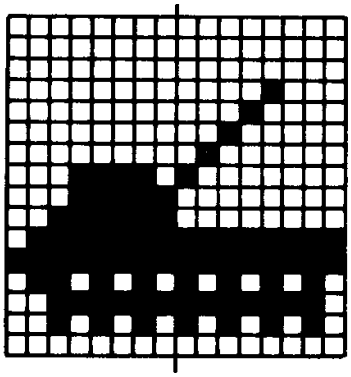
```
10  REM ufo
20  SCREEN 2,3:COLOR 15,1,1:CLS
30  FOR J=0 TO l
40   FOR I=1 TO 32
50      READ S:S$=S$+CHR$(S)
60   NEXT I
70   SPRITE$(J)=S$
80    SPRITES(J+1)=S$
90    S$=""
100 NEXT J
110 FOR I=1 TO 80
120   PUT SPRITE 1,(I*1.3,I),12,O
130   PUT SPRITE 0,(I*1.3,1),4,1
140 NEXT I
150 FOR P=l TO 50
160  PUT SPRITE 0,(I*1.3-1,I-2),9,2
170   FOR Q=l TO 50:NEXT Q
180   PUT SPRITE 0,(I*1.3-1,209),9,2
190   FOR Q=l TO 50:NEXT Q
200 NEXT P
210 RUN"CAS:" (210 GOTO 210)
8000 DATA 192,32,16,9,7,63,127,230,102,
          63,31,19,33,0,0,0
8010 DATA 3,4,8,144,224,252,254,103,
          102,252,248,200,132,0,0,0
8020 DATA 0,0,0,0,0,0,0,0,25,25,0,0,
          0,0,0,0
8030 DATA 0,0,0,0,0,0,0,0,152,152,0,
          0,0,0,0,0
```

De sprites komen tot leven. Het zijn in het algemeen nogal agressieve wezentjes. Soms zó agressief dat ze geen waardig sieraad zijn aan de tot nu toe opgebouwde parelketting. Er is echter een onderdeel aan een parelketting waarbij de zacht glanzende schoonheid van de parel geen rol speelt: het slot. Dat wordt meestal onzichtbaar weggemoffeld. Als sluitstuk van dit deel van het boek zullen we wat bruut geweld aan het vorige programma toevoegen waarin dan toch het onvermijdelijke destructieve element naar voren komt. Als excuus voor het geweld op het scherm kan worden aangevoerd dat het een goede manier is om ON SPRITE GOSUB te demonstreren. De computer kan zelf vaststellen of twee sprites met elkaar in botsing komen. We gaan dit proberen door de UFO die in het vorige programma voorkomt uit de lucht te schieten. In de eerste plaats moeten we hiervoor een tank het strijdtoneel op laten rijden.



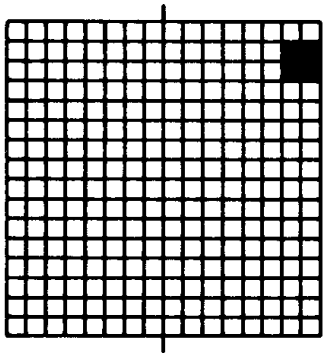
```
8040 DATA 0,0,0,0,0,0,0,30,63,63,127,255,
          106,63,42,0
8050 DATA 0,0,0,0,0,0,0,0,255,0,255,255,
          171,254,170,0
```

Als de gevechtspositie is ingenomen, richten we de loop op de UFO:



```
8060 DATA 0,0,0,0,0,0,0,30,63,63,127,255,
          106,63,42,0
8070 DATA 0,0,0,4,8,16,32,64,128,0,255,
          255,171,254,170,0
```

Nu nog het projectiel dat de loop op de juiste plaats verlaat:



```
8080 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
8090 DATA 0,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0
```

De strijd tegen de indringer kan beginnen. Let op regel 20. We hebben de ruimte nodig dus gaan we naar SCREEN 2,2.

```
10 REM attack
20 SCREEN 2,2:COLOR 15,1,1:CLS
22 ON SPRITE GOSUB 1000
30 ON SPRITE GOSUB 1000
40 FOR J=0 TO 4
50 FOR I=1 TO 32
60 READ S:S$=S$+CHR$(S)
70 NEXT I
80 SPRITE$(J)=S$
90 S$=""
100 NEXT J
110 FOR I=1 TO 70
120 PUT SPRITE 1,(I*3,I),12,0
130 NEXT I
140 FOR P=1 TO 10
150 PUT SPRITE 0,(I*3-3,I-2),9,1
160 FOR Q=1 TO 50:NEXT Q
170 PUT SPRITE 0,(I*3-3,209),9,1
180 FOR Q=1 TO 50:NEXT Q
190 NEXT P
200 FOR I=1 TO 100
210 PUT SPRITE 2,(I,170),3,2
220 FOR Q=1 TO 30:NEXT Q
230 NEXT I
240 FOR Q=1 TO 300:NEXT Q
250 PUT SPRITE 2,(I-1,170 ) ,3,3
260 SPRITE ON
270 FOR Q=1 TO 300:NEXT Q
280 FOR I=1 TO 100
290 PUT SPRITE 4,(I+100,170-I),11,4
300 NEXT I
302 SPRITE OFF
310 RUN"CAS:"          (310 GOTO 310)
1000 FOR I=1 TO 50
1010   CIRCLE(220,80),RND(1)*30,RND(1)*14
1020 NEXT I
1030 SPRITE OFF
1040 RETURN
8000 DATA 192,32,16,9,7,63,127,230,102,63,
          31,19,33,0,0,0
8010 DATA 3,4,8,144,224,252,254,103,102,
          252,248,200,132,0,0,0
8020 DATA 0,0,0,0,0,0,0,0,25,25,0,0,0,0,0,0
8030 DATA 0,0,0,0,0,0,0,0,152,152,0,0,0,0,0,0
8040 DATA 0,0,0,0,0,0,0,30,63,63,127,255,106,
          63,42,0
8050 DATA 0,0,0,0,0,0,0,0,255,0,255,255,171,
          254,170,0
```

```
8060 DATA 0,0,0,0,0,0,0,30,63,63,127,255,106,
          63,42,0
8070 DATA 0,0,0,4,8,16,32,64,128,0,255,255,
          171,254,170,0
8080 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
8090 DATA 0,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0
```

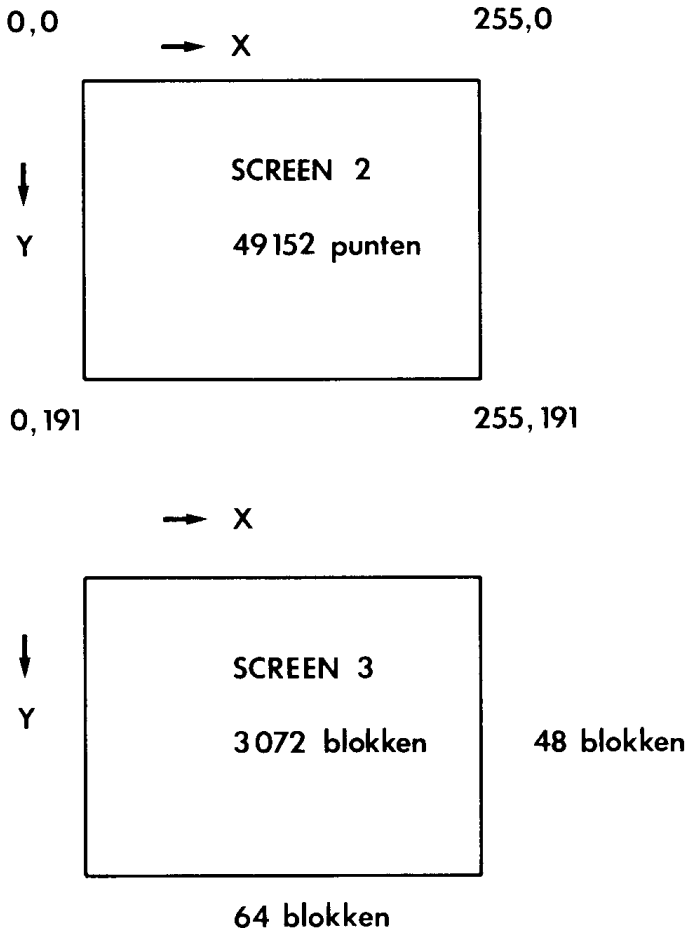
Wat de pareltjes aangaat: dit was het dan. Een parelsnoer vol interessante schermpjes. Sommige met beweging, een paar met geluid. Maar altijd een demonstratie van de onuitputtelijke grafische mogelijkheden van de MSX-computer. Een volwassen systeem binnen ieders bereik. Tussen het lezen van dit boek door heeft u veel typewerk verricht. Maar alles wat u heeft gedaan, ligt vast. Ga er eens rustig voor zitten en laat de parelketting nog eens langs het scherm glijden. Het is een indrukwekkend geheel geworden. Wellicht heeft u er al zelfgemaakte juweeltjes in opgenomen. Als het bandje de laatste parel heeft prijsgegeven, blijft het doelloos doorlopen. Wanhopig op zoek naar een volgend stukje werk. Uw werk !!!

17 Overzicht van de grafische commando's

SCREEN

```
SCREEN 2,Ø,1,Ø,Ø
       |  |  |  |
       |  |  |  Type printer
       |  |  SAVE-snelheid
       |  Toetsaanslag klik aan/uit
       Sprite-grootte
    Schermsoort
```

SCREEN 0 Tekstscherm 24 regels met 40 tekens.
SCREEN 1 Tekstscherm 24 regels met 32 tekens (groter lettertype).



Afbeelding 17-1 De opbouw van scherm 2 en scherm 3.

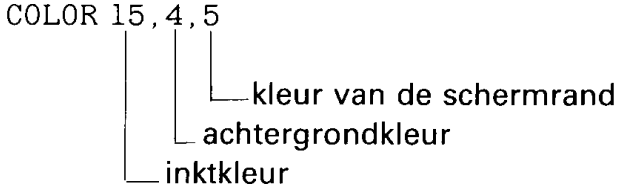
SCREEN 2 Scherm opgebouwd uit 192 horizontale rijen van 256 beeldpunten die onafhankelijk van elkaar kunnen worden aan- en uitgezet. Beperking: Per groepje van acht horizontale beeldpunten mogen niet meer dan twee kleuren voorkomen.

SCREEN 3 Scherm opgebouwd uit 192 horizontale rijen van 256 beeldpunten die in groepjes van 4x4 beeldpunten onafhankelijk van elkaar kunnen worden aan- en uitgezet.

SCREEN ,0 Sprite-afmeting 8x8 beeldpunten.
SCREEN ,1 Sprite-afmeting 8x8 beeldpunten echter vergroot weergegeven.
SCREEN ,2 Sprite-afmeting 16 x 16 beeldpunten.
SCREEN ,3 Sprite-afmeting 16 x 16 beeldpunten echter vergroot weergegeven.

De drie laatste SCREEN-variabelen worden zelden gebruikt.

COLOR




```
COLOR 15,4,5
```

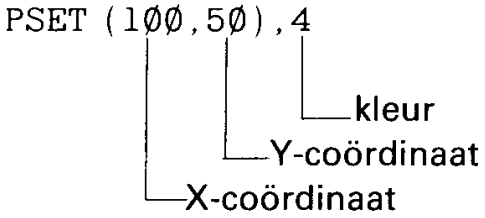
Geeft witte letters op een donkerblauwe achtergrond waarbij de schermrand lichtblauw is.
Een achtergrondkleur verandert pas na een CLS-opdracht.

Kleurnummer	Kleur
0	Transparant
1	Zwart
2	Groen
3	Lichtgroen
4	Donkerblauw
5	Lichtblauw
6	Donkerrood
7	Cyaan
8	Rood
9	Lichtrood
10	Donkergeel

Kleurnummer	Kleur
11	Lichtgeel
12	Donkergroen
13	Paars
14	Grijs
15	wit

PSET, PRESET

Werken uitsluitend bij SCREEN 2 en 3.




```
PSET   (100,50)
```

Laat het beeldpunt 100,50 oplichten in de geldende inktkleur.

```
PSET   (100,50),10
```

Laat het aangegeven beeldpunt oplichten in de kleur geel.

```
PSET  STEP(20,10),10
```

Laat een punt geel oplichten dat 20 punten rechts van en 10 punten onder het laatst getekende beeldpunt ligt.

PRESET dooft een punt. Als PRESET van een kleur is voorzien, is er geen verschil met de PSET-opdracht.

LINE

Werkt uitsluitend bij SCREEN 2 en 3.

```
LINE   (10,20)-(100,150)
```

Trekt een lijn vanaf de beeldcoördinaten 10,20 naar het punt 100,150 in de van kracht zijnde inktkleur.

`LINE   (10,20)-(100,150),6` Trekt dezelfde lijn in rood.

```
LINE   (10,20)-(100,150) ,,B
```

Tekent een rechthoek met de punten 10,20 en 100,150 als hoekpunten in de geldende inktkleur.

```
LINE   (10,20)-(100,150),12,BF
```

Tekent de rechthoek in de kleur donkergroen, maar vult hem bovendien op met de kleur donkergroen.

```
LINE  -STEP(20,20)
```

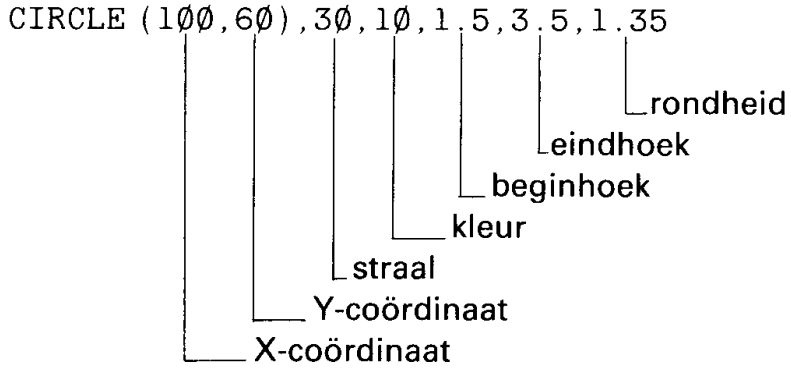
Trekt een lijn vanaf het laatst getekende punt naar een punt dat 20 plaatsen naar rechts en twintig plaatsen naar beneden ligt.

```
LINE  -STEP(20,20),6,BF
```

Tekent een donkerrood opgevuld vierkant met zijden van 20 x 20 beeldpunten. Het beginpunt is de laatst getekende beeldpunt.

CIRCLE

Werkt uitsluitend bij SCREEN 2 en 3.




```
CIRCLE   (100,60),30
```

Tekent een cirkel met als middelpunt 100,60 en met een straal van 30 beeldpunten in de geldende inktkleur.

```
CIRCLE   (100,60),30,10
```

Tekent een gele cirkel.

```
CIRCLE   (100,60),30,9,1.5,4.5
```

Tekent alleen de linkerhelft van de cirkel in de kleur lichtrood.

```
CIRCLE   (100,60),30,9,,,1.35
```

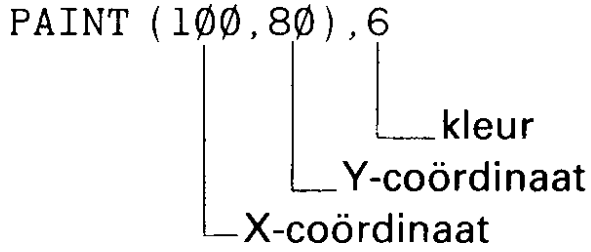
Tekent een ronde cirkel. Het getal kan per monitor verschillend zijn. In het ideale geval is het getal 1.
De rondheid van de cirkel duidt de verhouding tussen X en Y aan. X en Y zijn resp. de horizontale en de verticale as van de cirkel.

```
CIRCLE  STEP(20,-10),40
```

Tekent een cirkel waarbij het middelpunt 20 beeldpunten rechts van en 10 beeldpunten boven het laatst getekende beeldpunt ligt.

PAINT

Werkt uitsluitend bij SCREEN 2 en 3.




```
PAINT   (100,80)
```

Vult een in de geldende inktkleur getekende gesloten contour, waarbinnen het opgegeven punt ligt, op met de geldende inktkleur.

```
PAINT   (100,80),6
```

Vult een donkerrood getekende gesloten contour waarbinnen het opgegeven punt ligt donkerrood op.

```
PAINT STEP(-20,10),12
```

Gaat uit van een beeldpunt dat ten opzichte van het laatst getekende beeldpunt 20 plaatsen naar links en 10 plaatsen naar beneden is verschoven.

DRAW

Werkt uitsluitend bij SCREEN 2 en 3.

DRAW voert tekenopdrachten uit die worden gegeven in de vorm van een string.

```
DRAW   "U60R40"
```