ROBERT YOUNG

HET MSX

INSTRUCTIEBOEK

LEER GESTRUCTUREERD
PROGRAMMEREN MET
MUZIEK EN GRAFIEK



ADDISON-WESLEY

HET MSX INSTRUCTIEBOEK Robert Young

HET MSX INSTRUCTIEBOEK

LEER GESTRUCTUREERD PROGRAMMEREN MET MUZIEK EN GRAFIEK

HET MSX INSTRUCTIEBOEK

LEER GESTRUCTUREERD PROGRAMMEREN MET MUZIEK EN GRAFIEK

ROBERT YOUNG



ADDISON-WESLEY

Young, Robert

Het MSX instructieboek. Leer gestructureerd programmeren met muziek en grafiek / Robert Young; [vert. en bew. uit het Engels door LINE UP tekstprodukties]. – Amsterdam: Addison-Wesley Europe b.v.
Vert. en bew. van: Creative sound and graphics on your Commodore 64. – Wokingham: Addison-Wesley, 1984.
ISBN 90-6789-062-6
SISO 365.3 UDC 681.3.06
Trefw.: computerspelen; programmeren / MSX (computer); programmeren.

Omslagontwerp: BSE Advertising
Produktie en bewerking: LINE UP tekstprodukties
Illustraties: C.F.M. de Valk

© 1986 Addison-Wesley Europe b.v.

Niets uit deze uitgave mag worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm, geluidsband, elektronisch of op welke andere wijze ook en evenmin in een retrieval system worden opgeslagen zonder voorafgaande schriftelijke toestemming van de uitgever.

INHOUDSOPGAVE

Voorwoord		VII
Inleiding		IX
1	Het toetsenbord	1
2	In beweging	7
3	De VPOKE-instructie	11
4	De VPEEK-instructie	19
5	De sleutel tot het spel	23
6	We passen de stukjes in elkaar	31
7	De joystick	39
8	Strings	45
9	Volop actie	53
10	Grafiek uit eigen keuken	67
11	Reverse tekens	75
12	Een adempauze	77
13	Sprites	91
14	Meerkleuren-sprites	105
15	Sprite-animatie	109
16	Pret met sprites	111
17	Greep op geluid: SOUND	119
18	Meer geluid: PLAY	129
19	Grafiek, bit voor bit	137
20	Spelplezier	141
Bijlagen		165

Voorwoord

Voor programmeerdoeleinden is de MSX uitermate geschikt. Binnen enkele weken na aanschaf kon u vast al heel aardige programma's schrijven. Maar de nodige spanning en perfectie ontbraken waarschijnlijk nog. Het doel van dit boek is uw resultaten een professioneel tintje te geven. Een keur van praktische en makkelijk toepasbare technieken passeert de revue.

Geluidsroutines, bewegende grafische figuren, dramatische effecten - wat wilt u nog meer in dit boek van Robert Young, schrijver van meer dan tien populaire computerboeken. Naast een aantal routines voor eigen programma's bevat dit boek ook kant en klare programma's. U zult er veel plezier aan beleven.

Was de relatie met uw MSX de laatste tijd wat bekoeld, dan is dit boek het juiste recept. U moest er maar eens induiken.

Inleiding

Dit boek neemt u mee op ontdekkingsreis. De MSX is een veelzijdige computer met grafische mogelijkheden en een driestemmige synthesizer. Ik hoop dat de functies en programma's die aan de orde komen, bijdragen tot een degelijke kennis van wat uw computer op het terrein van grafiek en geluid te bieden heeft.

Dit is een boek voor iedereen, ook voor mensen die nog nooit een regel hebben geprogrammeerd. Het taalgebruik is eenvoudig gehouden zodat elke lezer de informatie en ideeën kan volgen. Lees het met de computer binnen handbereik. Droogzwemmen is niet de manier om de kneepjes onder de knie te krijgen. Pas als een programma loopt en de effecten zichtbaar worden, krijgt u er echt vat op. Een hoop demonstratieprogramma's en vijf schitterende spelen zorgen ervoor dat u zich niet zult vervelen. Het zijn uitstekende voorbeelden van wat er allemaal kan met een MSX.

Typ de programma's heel nauwkeurig in. Zelfs door een kleine fout kan een programma volledig in de soep draaien. Controleer alle tekens, zelfs het aantal spaties in de PRINT-instructies. Ze hebben vaak een functie bij de lay-out van het scherm.

Maak voor de zekerheid altijd eerst een kopie op band of schijf voordat u een programma RUNt. Dat geldt speciaal voor bewegende grafische figuren. Hele listings opnieuw intikken als er iets mis is, behoort dan tot het verleden.

De MSX geeft gewoonlijk vrij nauwkeurig aan waar de fout zit. Bijvoorbeeld bij SYNTAX ERROR vermeldt de computer om welke regel het gaat. Controleer die regel teken voor teken. Er staat iets dat niet als correcte BASIC-instructie kan worden geïnterpreteerd. Pak het boek er maar bij want waarschijnlijk heeft u een typfout gemaakt. Veel fouten kunnen worden vermeden als iemand de regels voorleest.

Het wordt nu tijd dat u de computer aanzet en het eerste hoofdstuk opslaat. Daar begint de reis door de wereld van sprites, zelfgemaakte grafische plaatjes, grafiek en de geluids-chip van de MSX.

Hoofdstuk 1

Het toetsenbord

We staan aan het begin van onze ontdekkingsreis door de wereld van geluid en grafiek. Daarom staan we eerst maar eens stil bij het toetsenbord van de MSX, de onmisbare schakel tussen theorie en praktijk. Achter het toetsenbord van de MSX gaat meer schuil dan de gebruikelijke alfabetische en numerieke tekens: er kunnen ook grafische tekens mee worden geproduceerd. De MSX heeft daarvoor een extra toets, de GRAPH-toets. Drukt u een toets tegelijk met deze toets in, dan verschijnt er een grafisch teken op het scherm. Een overzicht van alle karakters, vindt u in bijlage 1. We maken in dit boek gebruik van de mogelijkheid om (alfa)numerieke tekens samen met grafische tekens te gebruiken. Een voorbeeld daarvan is het volgende programma. Op het scherm verschijnt de mededeling 'IK BEN DE GROTE MSX' met een kader eromheen.

```
10 PRINT"[GR r,21*GR -,GR y]"
20 PRINT"[GR |] IK BEN DE GROTE MSX [GR |] "
30 PRINT"[GR v,21*GR -,GR n]"
```

In de listings in dit boek zijn de grafische tekens niet gePRINT, maar hebben we in verkorte vorm aangegeven hoe de tekens verkrijgbaar zijn. Ook in dit eerste programma is dat al het geval. Het grafische 'hoekje' krijgt u door tegelijkertijd de GRAPH-toets en de R in te drukken, het rechtopstaande streepje door tegelijkertijd GRAPH, SHIFT en | in te drukken. Indien het in een listing om afkortingen gaat, zijn deze tussen vierkante haken gezet.

Met een beetje fantasie kunt u met de grafische tekens figuren en tekeningen maken. Druk eens tegelijk op de GRAPH-toets en de F en daarna op GRAPH en Z. Druk tot slot op GRAPH-H. Op het scherm staan de contouren van een ruimteschip.

Galgje

Dit korte programma produceert een leuk plaatje van een gehangene. Het hoort bij het woordspelletje GALGJE en maakt gebruik van de grafische tekens die u krijgt door de GRAPH-toets te gebruiken. In het programma wordt veelvuldig gebruik gemaakt van de instructie LOCATE, maar daar gaan we later op in. In de REM-regels worden programma-onderdelen toegelicht toe. Voor een complete lijst van de bijzonder karakters en de wijze waarop u ze op het scherm kunt brengen, verwijzen we naar bijlage 1. Typ het programma nauwkeurig over en bekijk het resultaat na het met RUN te hebben gestart.

```
10 REM GALGJE
20 CLS:SCREEN 1
30 KEY OFF
40 LOCATE 5,5
50 PRINT"[GR r 5*GR -,GR y]"
60 LOCATE 5,6
70 PRINT"[GR |,SPATIE,GR /,3*SPATIE,GR |"
80 LOCATE 5,7
90 PRINT"[GR |,GR /,3*SPATIE,GR ["
100 LOCATE 5,8
110 PRINT"[GR |,4*SPATIE,GR r,GR -,GR y]"
120 LOCATE 5,9
130 PRINT"[GR |,3*SPATIE,GR /,GR |,SPATIE,GR
|,GR \]"
140 LOCATE 5,10
150 PRINT"[GR |,4*SPATIE,GR |,SPATIE,GR |]"
160 LOCATE 5,11
170 PRINT"[GR |,4*SPATIE,GR r,GR -,GR y]"
180 LOCATE 5,12
190 PRINT"[GR |,4*SPATIE,GR |,SPATIE,GR |]"
200 LOCATE 5,13
210 PRINT"[GR |,4*SPATIE,GR b,SPATIE,GR b]"
220 LOCATE 5,14
230 PRINT"[GR |]"
```

```
240 LOCATE 5,15
250 PRINT"[GR |]"
260 LOCATE 5,16
270 PRINT"[GR |]"
280 LOCATE 5,17
290 PRINT"[GR |]"
300 LOCATE 3,18
310 PRINT"[2*GR -,GR b,2*GR -]"
```

Kleur

De MSX stelt u een compleet kleurenpalet ter beschikking. De computer kent 15 verschillende kleuren en als extra mogelijkheid de 'kleur' transparant. Door middel van de instructie COLOR X,Y,Z kunt u gebruik maken van deze kleuren. U kunt respectievelijk de voorgrondkleur (X), de achtergrondkleur (Y) en de kleur van de beeldschermrand (Z) veranderen door waarden aan de verschillende variabelen toe te kennen. De tekens nemen automatisch de kleur van de voorgrond aan. COLOR 15,4,8 bijvoorbeeld resulteert in witte tekens, een donkerblauwe achtergrond en een rode beeldschermrand. Wanneer u de computer aanzet, zult u merken dat er geen beeldschermrand te zien is. Die rand is alleen zichtbaar in de SCREENs 1, 2 en 3, terwijl de MSX automatisch in SCREEN 0 opstart. Het is dan ook niet zinvol om in SCREEN 0 de variabele Z te definiëren.

Straks gaan we verder in op de mogelijkheden en eigenschappen van de verschillende SCREENs. En in hoofdstuk 3 bekijken wat de MSX nog meer met kleur kan doen.

De MSX beschikt over de volgende kleuren:

code	kleur
0	transparant
1	zwart
2	groen
3	lichtgroen
4	donkerblauw
5	lichtblauw
6	donkerrood
7	hemelsblauw
8	rood
9	lichtrood
10	donkergeel
11	lichtgeel
12	donkergroen
13	rose
14	grijs
15	wit

Het volgende programma vraagt naar de voorgrondkleur, de achtergrondkleur en de kleur van de beeldschermrand. Hebt u de waarden ingevuld, dan voert de computer uw wensen uit.

```
10 REM COLOR
20 SCREEN 1
30 KEY OFF
40 PRINT"KLEUREN TUSSEN 1 EN 15":PRINT
50 PRINT"WELKE VOORGRONDKLEUR";
60 INPUT X
70 PRINT"WELKE ACHTERGRONDKLEUR";
80 INPUT Y
90 PRINT"WELKE BEELDSCHERMRAND KLEUR";
100 INPUT Z
110 IF X>15 OR Y>15 OR Z>15 THEN CLS: GOTO 40
120 COLOR X,Y,Z
130 FOR TD=1 TO 1000:NEXT TD
140 CLS
150 GOTO 40
```

Krachtige hulpmiddelen

Naast de instructies die we hebben besproken, bent u in de voorgaande programma's enkele instructies tegengekomen waar we weinig over hebben verteld. Zo dient de instructie CLS om het hele scherm te wissen. Het is dan ook handig deze instructie aan het begin van een programma te zetten, zodat alles wat niet met het programma te maken heeft van het scherm verdwijnt. Een andere wisinstructie is de instructie KEY OFF. De balk met functietoetsen onder in beeld verdwijnt na gebruik van KEY OFF.

Door de instructies TAB(X) en LOCATE X,Y in samenhang met de instructie PRINT te gebruiken kunt u tekens op iedere gewenste plaats op het scherm PRINTen. In het volgende hoofdstuk vertellen we iets meer over beide instructies. Met behulp van CLS, LOCATE en TAB kunt u bijvoorbeeld een parcours maken voor een racespelletje:

```
10 CLS
20 SCREEN 0
30 X$="[5*GR p]"
40 Y=10.5
50 Q=RND(-TIME)
60 LOCATE Y,2
70 X=RND(1)
80 Y=Y+(X<.5)-(X>.5)
90 IF Y<5 THEN Y=5
100 IF Y>25 THEN Y=25
110 PRINT TAB(Y) X$
120 GOTO 70
```

SCREEN

Op de MSX kunt u vier verschillende SCREENs kiezen: twee tekstschermen en twee grafische schermen. Ieder scherm heeft zijn eigen mogelijkheden en onmogelijkheden. De tekstschermen dienen voor de weergave van letters, cijfers en grafische tekens. Op de beide grafische schermen kunt grafische voorstellingen als cirkels, lijnen en stippen afbeelden.

SCREEN 0
Dit is het eerste tekstscherm, waarop maximaal 40 tekens horizontaal en 24 lijnen verticaal passen. De beginbreedte (default) van

SCREEN 0 is 37 tekens. Voor de lettertekens zijn 6 punten beschikbaar, wat in het algemeen geen probleem is. De meeste tekens zijn 6 punten breed, op enkele grafische tekens na. Tekens die breder zijn dan 6 punten zijn dan ook niet in hun geheel weer te geven. Dit scherm verschijnt op de monitor wanneer u de computer inschakelt.

SCREEN 1
Het tweede tekstscherm, met maximaal 32 tekens horizontaal en 24 lijnen verticaal. De default-instelling van SCREEN 1 is 29 tekens. Voor de tekens zijn 8 punten beschikbaar, wat het scherm beter leesbaar maakt dan SCREEN 0. Dit scherm biedt de mogelijkheid tegelijkertijd gebruik te maken van zelf gedefinieerde karakters en van letters en bovendien de mogelijkheid om met meer dan twee kleuren te werken. Om deze redenen zullen we in het verloop van het boek meestal gebruik maken van SCREEN 1.

SCREEN 2
Dit is het eerste grafische scherm, met een resolutie van 256*192 pixels (beeldpunten). Het is het scherm met de meest gedetailleerde weergave omdat elke grafische voorstelling met 1 pixel tegelijk wordt afgebeeld.

SCREEN 3
Het tweede grafische scherm, met een resolutie van 256*192 pixels. Dit scherm is echter minder gedetailleerd dan het voorgaande omdat per grafische afbeelding 4 pixels worden gebruikt. Deze 4 pixels zijn nodig omdat dit scherm de mogelijkheid biedt tot meerkleurenweergave. De verdere eigenschappen van de verschillende schermen met betrekking tot het gebruik van sprites wordt uitgelegd in het hoofdstuk 'Sprites'.

Hoofdstuk 2

In beweging

In het vorige hoofdstuk hebben we het toetsenbord van de MSX onder de loep genomen en bekeken op welke wijze we met kleuren en grafische tekens kunnen werken. In dit hoofdstuk komen de beginselen van bewegende grafiek aan de orde en de verschillende manieren waarop figuren over het scherm kunnen worden bewogen.

Animatie

Voor we hiemee beginnen is het nuttig iets te vertellen over animatie in het algemeen. Het principe van animatie is te zien in tekenfilms, die in felle kleuren, snel en vaak humoristisch, de meest onwaarschijnlijke figuren en acties vertonen. Een tekenfilm is een lange reeks plaatjes die telkens iets van elkaar verschillen. Elk plaatje wordt een fractie van een seconde geprojecteerd en dan vervangen om de illusie van onafgebroken beweging te wekken. Beweging op de computer gaat op dezelfde manier. We zetten een teken op het scherm, laten het even staan en verwijderen het weer. Vervolgens kiezen we een andere positie, enzovoort. Wat we nodig hebben om dit uit te voeren is een efficiënte, snelle methode plus een manier om de beweging naar onze hand te zetten. Iets horizontaal over het scherm bewegen gaat het gemakkelijkst met PRINT-instructies: met de ene PRINT-instructie zetten we een teken ergens neer en met de volgende verplaatsen we het. Typ de volgende regels in en RUN ze.

```
10 REM BEWEEG STERRETJE
20 CLS:REM SCHERM SCHOON
30 KEY OFF
40 FOR X=1 TO 36
50 PRINT CHR$(11);:REM ZET CURSOR LINKSBOVEN
OP HET SCHERM
60 PRINT TAB(X)"* ":REM SPATIE STERRETJE
70 FOR TD=1 TO 50:NEXT TD
80 PRINT CHR$(11);:REM ZET CURSOR LINKSBOVEN
OP HET SCHERM
90 PRINT TAB(X)" ":REM SPATIE
100 NEXT X
```

Nadat dit kleine programma is geRUNd, verschijnt er een asterisk (*) linksboven op het scherm. Tijdens het programma beweegt deze naar rechts. We zullen eens zien hoe dit in zijn werk gaat.

Regel 10 is een REM-statement. REM-regels zijn bedoeld om informatie over het programma toe te voegen. De computer negeert dit soort regels. Regel 20 wist het scherm. In regel 40 begint de FOR/NEXT-lus die het verloop van het programma regelt. De breedte van het scherm is 37 tekens. We moeten de asterisk dus 37 keer PRINTen om hem van de ene naar de andere kant van het scherm te verplaatsen. Regel 60 zet de asterisk op het scherm. We zetten achter de PRINT-instructie een puntkomma. Zouden we dat niet doen dan zou het sterretje niet horizontaal maar verticaal over het scherm bewegen. De TAB in regel 60 is heel belangrijk. Voordat het sterretje wordt gePRINT, moeten er een aantal spaties komen te staan, gerekend vanaf de linkerkant van het scherm. Dat aantal moet steeds met 1 worden verhoogd, anders zou er van beweging geen sprake zijn. Om beweging te suggereren moet de laatste asterisk worden vervangen door een spatie. De waarde van X in de FOR/NEXT-lus bepaalt het aantal spaties. Hoe groter X des te verder naar rechts verschijnt de nieuwe asterisk. De FOR/NEXT-lus in regel 70 bepaalt de vertraging waardoor het sterretje kan worden waargenomen. U kunt de beweging versnellen of vertragen door de waarde 50 in regel 70 te veranderen. Voor echt flitsende actie dient u regel 70 helemaal weg te laten. Regel 90 is gelijk aan regel 60. Voordat het volgende sterretje verschijnt, PRINT deze regel een spatie op de plaats van de laatste. Regel 100 is het einde van de buitenste FOR/NEXT-lus. Het programma springt naar regel 40, waar X een nieuwe waarde krijgt.

Een andere manier

Het volgende programma doet hetzelfde op een iets andere manier. We maken geen gebruik van de instructie TAB maar van de wat krachtiger instructie LOCATE X,Y. Met LOCATE hebt u niet alleen de mogelijkheid om de X-waarde (voor de horizontale positie) aan te geven, maar ook de Y-waarde (voor de verticale positie). Het sterretje hoeft nu ook niet meer boven langs het scherm te bewegen, u kunt het op elke gewenste hoogte laten bewegen.

```
10 REM BEWEEG STERRETJE
20 CLS:REM SCHERM SCHOON
30 KEY OFF
40 FOR X=1 TO 36
50 LOCATE X,12:REM PLAATS CURSOR
60 PRINT "*":REM SPATIE STERRETJE
70 FOR TD=1 TO 50:NEXT TD
80 LOCATE X,12:REM PLAATS CURSOR
90 PRINT " ":REM SPATIE
100 NEXT X
```

Horizontaal bewegen is wel leuk, maar verticaal is ook nooit weg. Nu we de instructie LOCATE kennen, is verticaal bewegen een fluitje van een cent. We hoeven alleen in plaats van de variabele X, de variabele Y te verhogen.

```
10 REM BEWEEG STERRETJE
20 CLS:REM SCHERM SCHOON
30 KEY OFF
40 FOR Y=1 TO 22
50 LOCATE 18,Y:REM PLAATS CURSOR
60 PRINT "*":REM SPATIE STERRETJE
70 FOR TD=1 TO 50:NEXT TD
80 LOCATE 18,Y:REM PLAATS CURSOR
90 PRINT " ":REM SPATIE
100 NEXT Y
```

Het sterretje verscheen bovenaan het scherm, ongeveer midden in beeld en maakte toen een vloeiende beweging naar beneden. Het principe van een sterretje PRINTen en even later weer weghalen is hetzelfde als in de vorige programma's. Maar zoals gezegd wordt nu

door middel van de FOR/NEXT-lus in regel 40 de waarde van Y steeds verhoogd. Regel 70 bepaalt weer de vertraging waardoor het sterretje kan worden waargenomen.

Met PRINT horizontale en verticale bewegingen realiseren is wel aardig, maar veel te omslachtig. Bovendien kleeft er het bezwaar aan dat we de beweging niet op elk moment in de hand hebben. Bij computerspelen is dat zoals u weet absoluut noodzakelijk.

Een stukje proefPOKEn

Om bewegende figuren op het scherm te manipuleren, zijn VPOKE-instructies veel handiger dan PRINT-instructies. Met VPOKE (video-POKE) kunnen we direct waarden in het video-RAM van het computergeheugen zetten. We zullen met behulp van de VPOKE-instructie de asterisk nogmaals over het scherm laten bewegen. De vele mogelijkheden van VPOKE en alle toeters en bellen worden in het volgende hoofdstuk uitgebreid besproken. We zullen het onderstaande programma dan ook niet helemaal ontleden.

```
10 REM BEWEEG STERRETJE
20 CLS:REM SCHERM SCHOON
30 KEY OFF
40 FOR P=440 TO 479
50 VPOKE P,42
60 FOR TD=1 TO 50:NEXT TD
70 VPOKE P,32
80 NEXT P
```

Hoofdstuk 3

De VPOKE-instructie

Op de vorige bladzijden is nog niet duidelijk geworden hoe VPOKE-instructies kunnen worden gebruikt om figuren over het scherm te bewegen. Het is zo langzamerhand tijd dat we daar wat aandacht aan besteden. Het beeldscherm van de MSX bestaat in SCREEN 0 uit 40 kolommen en 24 regels. Op één regel passen dus 40 tekens. Een kolom wordt aangegeven als X-waarde en een regel als Y-waarde. Een snelle rekensom leert dat er in totaal 960 (40*24) posities zijn. Elke positie heeft zijn eigen geheugenadres en elk teken zijn eigen waarde. Een overzicht van de waarden van alle tekens staat in een bijlage achterin het boek.
Wanneer u de VPOKE-instructie gebruikt in SCREEN 1, moet u er rekening mee houden dat het scherm niet uit 40 maar uit 32 kolommen bestaat. Verder begint het schermgeheugen niet op adres 0, zoals in SCREEN 0, maar op adres 6144.

Positie

Zoals vermeld beginnen de schermgeheugens van de verschillende SCREENs op verschillende plaatsen. De positie linksboven in SCREEN 0 is 0 en de positie linksboven in SCREEN 1 is 6144. Een A linksboven in SCREEN 0 krijgt u door het volgende in te typen:

```
SCREEN 0 ‹RETURN›
VPOKE 0,65 ‹RETURN›
```

Dezelfde A linksboven in SCREEN 1 krijgt u door te typen:

```
SCREEN 1 ‹RETURN›
VPOKE 6144,65 ‹RETURN›
```

Om beweging te realiseren moeten we het teken overal op het scherm kunnen volgen. De volgende formule regelt dat:

VPOKE (beginadres) + X + Y * (schermbreedte)

De variabele 'beginadres' is het begin van het schermgeheugen in het betreffende SCREEN en daarom een constante. De variabele X geeft de kolom aan waar het teken moet komen te staan. X kan een waarde tussen de 0 en 39 krijgen, want een regel kan maximaal 40 tekens breed zijn (SCREEN 0). De Y in de formule staat voor de regel waarop we het teken willen hebben. Het scherm bestaat uit 24 regels, dus kan Y tussen 0 en 23 liggen. X en Y zijn de coördinaten van een positie op het scherm. Met deze formule kunnen we de A opnieuw op het scherm laten verschijnen. Wis het scherm, typ de volgende regels in en RUN ze.

```
10 SCREEN 1
20 VPOKE 6144+16+12*32,65
```

Als het goed is prijkt de A midden op het scherm.

Regel 20 begint met 6144, het eerste adres van het schermgeheugen in SCREEN 1. De A moet midden op het scherm komen, dus vanuit de linker bovenhoek 20 stappen naar rechts (X-richting) en 12 naar beneden (Y-richting). Eén regel naar beneden betekent in SCREEN 1 32 plaatsen verder in het geheugen. Om de A 12 regels naar beneden te krijgen, moeten we 12 dus met 32 vermenigvuldigen.
Met de X- en Y-coördinaten kunnen we overal op het scherm een teken neerzetten. Op de beweging na is het zaakje compleet. Als we de formule in een lus plaatsen, aan X en Y beginwaarden toekennen en die in de lus veranderen, zijn we met onze bewegende grafische figuren al een aardig eind op de goede weg. Het teken van richting veranderen als het de rand van het scherm bereikt, is het laatste wat er dan nog aan schort.

VPOKE 8192+...

Bij de instructie VPOKE geeft u in SCREEN 1 en 2 nog een extra optie, namelijk kleur. Met behulp van VPOKE kunt u in de adressen 8192-8224 de voor- en achtergrondkleur van karakters veranderen. VPO-

KEn in deze adressen is op een verfijnde manier gebruik maken van de instructie COLOR. U kunt in plaats van alle karakters in één keer telkens 8 tekens tegelijk veranderen: adres 8192 voor de karakters 0-7, 8193 voor de karakters 8-15, enzovoort. Om met succes kleuren te kunnen veranderen moet u enig begrip hebben van binaire getallen. Elk adres bestaat uit één byte en elke byte bestaat uit 8 bits, oftewel twee nibbles (een nibble is een reeks van 4 bits). Nibble 1 (bit 0-3) geeft de achtergrondkleur aan en nibble 2 (bit 4-7) de voorgrondkleur. Hieronder ziet u op welke wijze de bits worden genummerd. Elke bit kan de waarde 0 of 1 krijgen.

bit 7 6 5 4 3 2 1 0

Omdat een nibble uit 4 bits bestaat en we met nullen en enen werken, zijn er 16 combinatiemogelijkheden. (2 tot de macht 4). Dat komt overeen met de zestien kleuren die u op de MSX ter beschikking staan.
Wilt u voor een karakter als achtergondkleur wit kiezen (decimaal 15), dan moet u in nibble 1 alle bits zetten. Binair 1111 komt overeen met decimaal 15. Wilt u de voorgrondkleur van de karakter lichtrood hebben (decimaal 9), dan zet u in nibble 2 de bits 4 en 7 wel en 5 en 6 niet. Binair 1001 komt overeen met decimaal 9. De getallen 0-16 worden binair als volgt geschreven.

decimaal	binair
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Hebt u beide kleuren bepaald, dan hoeft u alleen nog beide nibbles

te koppelen en het decimale getal uit te rekenen. Het koppelen is heel eenvoudig: u zet beide nibbles achter elkaar. De volgorde is: eerst nibble 2 en dan nibble 1. In ons voorbeeld ziet het er als volgt uit: 10011111.
Dit binaire getal omzetten in een decimaal getal is een fluitje van een cent, dat doet de MSX namelijk voor u. Wat u moet doen is aangeven dat u een binair getal wilt PRINTen. Daarvoor dient het voorvoegsel &B. Voer de volgende regel in:

PRINT &B10011111

Resultaat: 159

Daarmee weten we met welk getal we een lichtrood karakter op een witte achtergrond kunnen zetten. Nu nog het karakter uitzoeken dat we willen kleuren. De MSX beschikt over 256 karakters en zoals gezegd kunnen per adres elke keer 8 karakters tegelijk van kleur worden veranderd. Wilt u karakter 65 kleuren, dan moet u de 8e groep van 8 karakters van kleur veranderen (adres 8192+8 verandert de karakters 64-71 van kleur).

Voer in

SCREEN 1 ‹RETURN›
PRINT CHR$(65) ‹RETURN›
VPOKE 8192+8,159 ‹RETURN›

U ziet een lichtrode A op een witte achtergrond. Deze mogelijkheid om verschillende tekens verschillende kleuren te geven komt ons uitstekend van pas bij het maken van spelletjes en om iéts op het scherm te accentueren. In het programma HAPPY HAPPER hebben we veelvuldig gebruik gemaakt van deze mogelijkheid.
Gebruikt u na een VPOKE de instructie COLOR X,Y,Z, dan nemen alle karakters weer hun normale kleur aan.

Aan de rol

Allereerst een kort programma om u met lussen en beweging vertrouwd te maken. Er verschijnt een bal (GRAPH 0) die schuin over het scherm rolt.

```
10 SCREEN 1
20 CLS
30 X=2:Y=2
40 FOR L=1 TO 23
50 VPOKE 6144+X+Y*32,9
60 VPOKE 8193,20
70 FOR TD=1 TO 50:NEXT TD
80 VPOKE 6144+X+Y*32,32
90 X=X+1:Y=Y+1
100 NEXT L
110 END
```

In regel 30 krijgen X en Y beide de waarde 2. Dat is de plaats waar de bal het eerst verschijnt. In regel 40 begint de FOR/NEXT-lus. Regel 50 gebruikt de formule om de bal (waarde 9) op het scherm te VPOKEn. Met de nieuwe VPOKE 8192... kleuren we de bal zwart. Een korte vertraging (regel 70) zorgt ervoor dat de bal zichtbaar wordt. De formule in regel 80 vervangt de bal door een spatie (waarde 32). Regel 90 verhoogt de variabelen X en Y met 1 en tot slot (regel 100) keert het programma terug naar het begin van de lus.

Skippy-bal

In het volgende programma laten we de bal naar beneden vallen en in een andere richting weer omhoog stuiteren. Typ het programma in en RUN het.

```
10 REM SKIPPY-BAL
20 GOSUB 2000
90 FOR TD=1 TO 100:NEXT TD
100 VPOKE 6144+BX+BY*32,32
110 BX=BX+X
120 IF BX>31 OR BX<0 THEN GOSUB 1000
130 BY=BY+Y
140 IF BY>23 OR BY<0 THEN GOSUB 1100
150 VPOKE 6144+BX+BY*32,9
160 VPOKE K+1,20
180 FOR TD=1 TO 15:NEXT TD
200 GOTO 100
1000 X=-X
1010 BX=BX+X
```

```
1020 RETURN
1100 Y=-Y
1110 BY=BY+Y
1120 RETURN
2000 REM INITIALISEREN
2010 K=8192
2020 CLS
2030 SCREEN 1
2040 COLOR 15,4,11
2050 BX=5:BY=5
2060 X=1:Y=1
2070 RETURN
```

Regel 20 stuurt het programma naar een subroutine (regel 2000 e.v.), die de waarden initialiseert die in het programma een rol spelen. Regel 2020 wist het scherm. BX en BY staan voor *'bal in de X-richting'* en *'bal in de Y-richting'*. Regel 2060 voert X en Y in en geeft beide de waarde 1. Door ze bij BX en BY op te tellen beweegt de bal. Andere getallen in X en Y betekenen een andere richting. De RETURN (regel 2070) brengt ons terug in het hoofdprogramma. De lus die de beweging stuurt, begint in regel 100. Eerst wordt er een spatie geVPOKEt op de plaats waar de bal moet verschijnen. Op die manier kunnen we er zeker van zijn dat die positie leeg is. U zult later zien waarom dat nodig is. X wordt bij BX opgeteld (regel 110) om de bal een stapje naar rechts te bewegen. Regel 120 controleert of BX niet groter is dan 32 of kleiner dan 0, de randen van het scherm in SCREEN 0. Zijn de grenzen overschreden, dan springt het programma naar de subroutine in regel 1000; daar wordt X negatief gemaakt (-1).

Vervolgens telt regel 1010 X bij BX op, wat neerkomt op aftrekken omdat X negatief is. Was de rechterkant van het scherm bereikt (BX=32), dan is BX na 1010 gelijk aan 32. Bij de volgende keer dat het programma de lus doorloopt, wordt er weer 1 van BX afgetrokken (regel 110): de bal beweegt van rechts naar links. Als de bal de linkerkant van het scherm bereikt, laat regel 1000 hem weer de andere kant opgaan. We zagen al dat regel 1000 de variabele X omzet in een positief getal als deze een negatieve waarde had. De regels 130 en 140, met de subroutine in regel 1100, doen hetzelfde voor de Y-richting. Wanneer zowel de X-waarde als de Y-waarde zijn getest en uitgewerkt, wordt de bal op het scherm geVPOKEt. Regel 180 zorgt voor een korte vertraging en regel 200 brengt ons terug naar het begin van de lus. We hebben deze keer geen FOR/NEXT-lus gebruikt, omdat het programma onafgebroken moet lopen. De bal blijft stuiteren totdat u CTRL-STOP indrukt.

In dit programma is een elementaire methode gebruikt om de bal op het scherm te houden. Het is belangrijk zeker te weten dat de stuiterende bal niet het scherm uitvliegt. Als dat wel gebeurt, kunnen er problemen ontstaan. Komt de bal buiten de 960 schermadressen, dan kan het zijn dat u een waarde VPOKEt in geheugenplaatsen die een andere functie hebben. Gevolg: het toetsenbord 'hangt', het programma gaat verloren of een andere vervelende situatie treedt op. Als u uw eigen programma's voor bewegende grafiek schrijft, zorg dan in ieder geval voor veiligheidsmaatregelen om te voorkomen dat bewegende objecten het scherm verlaten. Maak altijd een kopie van het programma voor u het RUNt. Mocht er iets mis gaan, dan hoeft u niet alles opnieuw te programmeren.

GeVPOKEte tekens

Voordat we verder gaan een kort programma dat tekens op het scherm afbeeldt die u erop VPOKEt met de waarden 32 tot 95. Het gaat om de alfanumerieke karakterset.

```
10 REM GEPOKTE TEKENS
20 SCREEN 0
30 Y=120
40 KEY OFF
50 FOR X=32 TO 95
60 VPOKE Y,X
70 Y=Y+2
80 NEXT X
90 GOTO 90
```

Hoofdstuk 4

De VPEEK-instructie

We hebben gezien hoe we een bewegend object kunnen beletten het scherm te verlaten door te testen of de X- en Y-waarden binnen bepaalde grenzen liggen. Deze methode is erg beperkt en bovendien weten we niet of er botsingen op het scherm plaatsvinden. Daarvoor gebruiken we VPEEK, de tegenhanger van VPOKE.

We weten al dat we met VPOKE een waarde rechtstreeks in een geheugenplaats kunnen *zetten*. Met VPEEK kunnen we in een geheugenplaats *kijken* om te zien welke waarde zich daar bevindt. Daardoor kunnen we voor de bal uit kijken en zien waar deze tegenop botst. In het vorige programma kregen we een spatie door de waarde 32 in een adres te VPOKEn. Het is niet verwonderlijk dat VPEEK, gevolgd door een leeg adres, in 32 resulteert. Elke andere waarde betekent dat er wel iets staat. Laten we dit eens nagaan voor een concreet geval. Wis het scherm en toets de volgende regel in:

```
PRINT VPEEK(459)
```

Sluit de invoer af met RETURN. Als het midden van het scherm leeg is op het moment van invoer, verschijnt het getal 32. De instructie PRINT is nodig omdat de computer anders niets met deze regel zou doen. Bij VPEEK moet de geheugenplaats die u wilt bekijken altijd tussen haakjes staan. Evenals bij VPOKE is het bij VPEEK ook van belang in welk SCREEN u zich bevindt. Is dat SCREEN 0, dan is het beginadres 0, maar bevindt u zich in SCREEN 1 dan is het beginadres 6144.

Ga nu terug naar de programmaregels die in SCREEN 1 een A in geheugenlocatie 367 zetten met behulp van VPOKE. Zet de A weer op het scherm en voer in

PRINT VPEEK(6144+367)

Als antwoord krijgt u 65, de ASCII-code voor de letter A. ASCII staat voor *American Standard Code for Information Interchange*.

We breiden nu het programma Skippy-bal uit om te zien hoe we met VPEEK de bal kunnen beletten het scherm te verlaten. Typ deze nieuwe versie in en RUN 'm.

```
10 REM SKIPPY-BAL
20 GOSUB 2000
30 GOSUB 3000
100 VPOKE 6144+BX+BY*32,32
110 BX=BX+X
120 IF VPEEK(6144+BX+BY*32)=219 THEN GOSUB 10
00
130 BY=BY+Y
140 IF VPEEK(6144+BX+BY*32)=219 THEN GOSUB 11
00
150 VPOKE 6144+BX+BY*32,9
160 VPOKE K+1,20
180 FOR TD=1 TO 15:NEXT TD
200 GOTO 100
1000 X=-X
1010 BX=BX+X
1020 RETURN
1100 Y=-Y
1110 BY=BY+Y
1120 RETURN
2000 REM INITIALISEREN
2010 K=8192
2020 CLS
2030 SCREEN 1
2040 COLOR 15,4,11
2050 BX=5:BY=5
2060 X=1:Y=1
2070 RETURN
3000 FOR SX=6144 TO 6144+31
3010 VPOKE K+27,128
3020 VPOKE SX,219
3030 NEXT SX
3040 FOR SY=6880 TO 6880+31
3050 VPOKE SY,219
```

```
3060 NEXT SY
3070 FOR SY=6144 TO 6880 STEP 32
3080 VPOKE SY,219
3090 NEXT SY
3100 FOR YY=6175 TO 6911 STEP 32
3110 VPOKE YY,219
3120 NEXT YY
3130 RETURN
```

Regel 20 stuurt ons weer naar een initialisatieroutine. Regel 2020 wist het scherm. De startpositie van de bal staat in regel 2050. Daarna worden de variabelen X en Y ingesteld (regel 2060), die op precies dezelfde manier worden gebruikt als in het vorige programma.

Botsingen opsporen

Hier vinden we het eerste grote verschil. Regel 30 stuurt ons naar een nieuwe routine, die begint op regel 3000. Hier wordt een reeks FOR/NEXT-lussen en VPOKE-instructies gebruikt om een rand rondom het scherm te tekenen. Met regel 3130 keren we terug in het hoofdprogramma. Het andere verschil in dit deel van het programma is te vinden in de regels 120 en 140. In plaats van de waarden van BX en BY te testen voordat de bal op het scherm wordt geVPOKEt, gebruiken we VPEEK om te zien of de bal tegen een ruitje botst. Zo kunt u botsingen opsporen met elk teken. De rest van de hoofdlus werkt net als in het vorige programma.

VPEEK is een handige functie. Botsingen opsporen zonder het hoofdprogramma ingrijpend te wijzigen behoort nu tot de mogelijkheden. U hoeft alleen de volgende regels aan de nieuwe Skippy-bal-versie toe te voegen, en de regels 120 en 140 iets te veranderen.

```
120 IF VPEEK(6144+BX+BY*32)<>32 THEN GOSUB 10
00
140 IF VPEEK(6144+BX+BY*32)<>32 THEN GOSUB 11
00
3130 VPOKE K,244
3140 FOR RB=1 TO 15
3150 XX=INT(RND(-TIME)*960+1)
3160 IF VPEEK(6144+XX)=219 THEN GOTO 3140
3170 VPOKE 6144+XX,4
3180 NEXT RB
3190 RETURN
```

Deze regels verspreiden willekeurig witte ruiten over het scherm. Als dit programma wordt uitgevoerd, stuitert de bal niet langer in een regelmatig patroon over het scherm. Hij botst tegen ieder voorwerp dat in zijn baan ligt. Dit programma illustreert het principe van random (= willekeurige) beweging.

U weet nu genoeg over bewegende grafiek en hoe u die kunt gebruiken op de MSX. In het volgende hoofdstuk zullen we zien wat we met deze kennis kunnen doen.

Hoofdstuk 5

De sleutel tot het spel

Een teken op het scherm VPOKEn en het laten stuiteren levert nu geen problemen meer op. We hebben kennisgemaakt met de principes van de bewegende grafiek. Maar een bruikbaar spel hebben we nog steeds niet. Het belangrijkste ingrediënt voor bijna alle soorten bewegende grafiek is een object dat over het scherm beweegt en wordt bestuurd via toetsenbord of joystick. Dat kan van alles zijn: een mannetje dat stuiterballen moet ontwijken, een laserwapen uit Star Wars of een schildpad die over moet steken, enzovoort.

We bekijken eerst hoe we een teken heen en weer over het scherm kunnen bewegen terwijl het steeds op dezelfde regel blijft. Deze beweging is nodig voor het geschut in een schietspelletje of een racket bij een partijtje squash. Het volgende korte programma POKEt een X in het midden van een regel onder aan het scherm. Elke keer als u de A indrukt, beweegt het teken één plaats naar links. Drukt u op L, dan beweegt het teken één plaats naar rechts. Neem dit programma nauwkeurig over en speel er een tijdje mee. Daarna bespreken we hoe het werkt.

```
10 REM BEWEGEN MET BEHULP VAN HET TOETSENBORD
20 SCREEN 0
30 CLS
40 X=20
100 A$=INKEY$
110 IF A$="A" THEN X=X-1
120 IF A$="L" THEN X=X+1
130 IF X<0 THEN X=0
```

```
140 IF X>39 THEN X=39
150 VPOKE 480+X,88
160 VPOKE 480+X-1,32:VPOKE 480+X+1,32
170 GOTO 100
```

Regel 30 wist het scherm zodat we kunnen zien wat er gebeurt. In regel 40 krijgt variabele X een beginwaarde die dient om het teken over het scherm te bewegen. De beweging vindt plaats in de hoofdlus (regel 100 e.v.). De INKEY$-instructie leest de invoer van het toetsenbord. Omdat we letters gebruiken, moeten we een stringwaarde invoeren. Die wordt toegekend aan de variabele A$. Regel 110 test de waarde van A$. Als deze gelijk is aan A, wordt X met 1 verminderd. Regel 120 test eveneens de waarde van A$. Als deze gelijk is aan L, wordt de waarde van X met 1 verhoogd. Een andere toets heeft geen invloed op de waarde van X.

Beveiliging

Regel 130 en 140 houden het teken op dezelfde regel. Het teken beweegt naar een van beide zijkanten van het scherm en stopt daar. Regel 150 en 160 vervullen de belangrijkste functie in het programma. Deze regels bevatten een soortgelijke formule als in het programma Skippy-bal. Deze formule gebruikt als beginpunt schermpositie 480. Die bevindt zich in het midden van het scherm, drie regels van onderen. We tellen vervolgens de waarde van X bij deze plaats op en krijgen zo een nieuwe schermpositie, waarin we de waarde 88 (een X) VPOKEn. Regel 160 VPOKEt een spatie aan beide kanten van de actuele positie om er zeker van te zijn dat de X is verwijderd die tijdens de vorige lusdoorloop op het scherm was geplaatst. Als we dit niet zouden doen, zouden we tijdens het bewegen een spoor van tekens op het scherm achterlaten. Regel 170 stuurt het programma weer naar het begin van de lus.

Deze methode is aardig om heen en weer over een regel te bewegen, maar wat doen we als we het hele scherm onveilig willen maken? Voor de gemakkelijkste manier moeten we terug naar de formule voor de skippy-bal. Het enige grote verschil is dat we de X- en Y-waarden van het voorwerp veranderen via het toetsenbord in plaats van bij deze waarden 1 op te tellen of af te trekken. Voer het volgende programma in en RUN het. Met de volgende toetsen kunt u de X over het hele scherm bewegen: A (naar boven), S (naar beneden), K (naar links) en L (naar rechts).

```
10 REM BEWEGEN MET BEHULP VAN HET TOETSENBORD
20 SCREEN 0
30 CLS
40 XA=20:XD=12
50 WA=XA:WD=XD
60 A$=INKEY$
70 IF A$="A" THEN XD=XD-1
80 IF A$="S" THEN XD=XD+1
90 IF A$="K" THEN XA=XA-1
100 IF A$="L" THEN XA=XA+1
110 IF XA<1 THEN XA=1
120 IF XA>39 THEN XA=39
130 IF XD<1 THEN XD=1
140 IF XD>23 THEN XD=23
150 VPOKE XA+XD*40,88
160 IF A$<>"" THEN VPOKE WA+WD*40,32:GOTO 50
170 GOTO 60
```

Nadat het scherm in regel 30 is gewist, worden de beginwaarden toegekend aan de twee variabelen XA en XD, die we gebruiken om het teken op het scherm te VPOKEn. XA zorgt voor de horizontale beweging, XD voor de verticale. In regel 50 voeren we nog twee variabelen in, WA en WD. Deze worden verderop in het programma gebruikt om de oorspronkelijke plaats van het geVPOKEte teken te wissen. Voorlopig bevatten ze de actuele waarde van XA en XD. Het toetsenbord wordt weer met INKEY$ uitgelezen. In regel 70-100 wordt deze invoer getest. Gaat het om een A dan wordt variabele XD met 1 verminderd. Dit heeft als effect dat dat de X één plaats omhoog gaat. Bij S wordt XD met 1 verhoogd. Daardoor gaat de X één plaats omlaag. Is het een K, dan wordt variabele XA met 1 verlaagd om de X één plaats naar links te bewegen. Een L tenslotte verhoogt XA met 1 waardoor de X één plaats naar rechts verschuift. De regels 110-140 zorgen ervoor dat de X op het scherm blijft. In regel 160 staan WA en WD, die worden gebruikt om een spatie (waarde 32) te POKEn op de oude plaats. Die werd gedefinieerd door XA en XD, voordat deze variabelen werden gewijzigd in regel 70-100. Regel 150 VPOKEt de X (waarde 88) op het scherm. Met GOTO gaan we terug naar het begin van de bewegingslus (regel 170).

Bestudeer deze uitleg en het programma tot u er zeker van bent dat u het principe begrijpt. Als u deze routine en die van de stuiterende bal onder de knie hebt, is uw basiskennis van bewegende grafiek voldoende. U weet dan genoeg om met eigen programma's te begin-

nen. Een teken over het scherm bewegen via het toetsenbord is nu geen probleem meer. U hoeft zich vanzelfsprekend niet te beperken tot de toetsen uit de voorbeelden. Iedere toets is beschikbaar. Het enige dat moet gebeuren is de letter tussen aanhalingstekens veranderen in de gewenste tekens.

Blijf in beweging

Tot nu toe moesten we elke keer een toets indrukken om een teken van plaats te laten veranderen. Spelletjes zijn veel opwindender als tekens moeiteloos over het scherm schieten. In het voorgaande programma is dat nog niet het geval. Een methode om bewegingen vloeiend op het scherm te toveren zou heel welkom zijn. En die is er. Er zijn twee manieren om vloeiende beweging te realiseren. Welke we gebruiken is afhankelijk van het type spel. De eerste methode is een toets indrukken, waarna het teken op het scherm begint te bewegen en niet stopt of van richting verandert voor u een andere toets indrukt. Deze manier is bijzonder geschikt voor doolhofspelletjes, waarbij behendigheid en reactievermogen op de proef worden gesteld.

We zullen dit eens in praktijk brengen door het vorige programma op enkele plaatsen te wijzigen. Typ de volgende nieuwe versie in of verander de oude.

```
10 REM BEWEGEN MET BEHULP VAN HET TOETSENBORD
20 SCREEN 0
30 CLS
40 XA=20:XD=12
50 WA=XA:WD=XD
60 A$=INKEY$
65 IF A$<>"A" AND A$<>"S" AND A$<>"K" AND A$<
>"L" THEN A$=B$
67 B$=A$
70 IF A$="A" THEN XD=XD-1
80 IF A$="S" THEN XD=XD+1
90 IF A$="K" THEN XA=XA-1
100 IF A$="L" THEN XA=XA+1
110 IF XA<1 THEN XA=1
120 IF XA>39 THEN XA=39
130 IF XD<1 THEN XD=1
140 IF XD>23 THEN XD=23
```

```
150 VPOKE XA+XD*40,88
160 VPOKE WA+WD*40,32
170 GOTO 50
```

Als u er zeker van bent dat uw programma precies overeenstemt met onze versie, RUN het dan en druk daarna op de K. In plaats van één plaats naar links te schuiven schiet de X die kant uit. Als u geen andere toets indrukt, botst hij tegen de linker rand van het scherm. Deze verandering wordt veroorzaakt door de toevoeging van de regels 65 en 67. Laten we die eens nader bekijken. We nemen aan dat u zojuist het programma hebt geRUNd, maar nog geen toets hebt ingedrukt. De X bevindt zich midden op het scherm. Het programma komt in regel 60, er is geen invoer via het toetsenbord, dus A$ heeft geen waarde. Nu komen we bij een van de nieuwe regels. Regel 65 test of A$ gelijk is aan een van de besturingstoetsen. Hier is dat niet het geval, daarom wordt de waarde van B$ toegekend aan A$. Maar B$ heeft ook nog geen waarde. Daar waren we nog niet aan toe. In regel 67 wordt B$ gelijkgesteld aan de waarde van A$. Geen van beide heeft een waarde, dus in de regels 70-100 gebeurt niets. De X blijft waar hij is.

Heen en weer

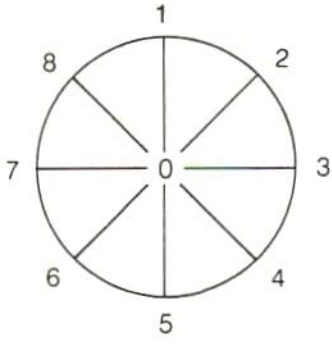
Druk nu op L. Bij de volgende lusdoorloop heeft regel 60 wel wat te doen. Hij kent de waarde L toe aan A$. Testregel 65 stelt vast dat A$ gelijk is aan een besturingstoets. Daarom wordt de waarde van A$ toegekend aan B$. Ook B$ is nu gelijk aan L. Het programma gaat verder in de regels die de variabelen veranderen en de X op het scherm beweegt één plaats naar rechts. Het programma keert terug naar regel 50. Deze keer is er geen invoer, omdat u niet meer op de toets drukt. Omdat er geen invoer is, kent de INKEY$-instructie geen waarde toe aan A$. Testregel 65 stelt vast dat A$ niet gelijk is aan een besturingstoets. Daarom wordt A$ gelijkgesteld aan B$, die nog steeds de waarde L bevat. Op die manier komt er een L in variabele A$, zonder dat een toets hoeft te worden ingedrukt. Handig, nietwaar? Door twee regeltjes in te voegen kunt u een programma ingrijpend veranderen.

Het alternatief

Inmiddels zult u zich wel hebben afgevraagd waarom we nog geen gebruik hebben gemaakt van de cursorbesturingstoetsen die rechts

op de computer zitten. We noemen die hier het alternatief, maar u zult merken dat we voor beweging in het verloop van het boek alleen nog maar gebruik zullen maken van deze cursorbesturingstoetsen of van een joystick, en niet meer van de alfanumerieke toetsen van de computer. Het principe blijft gelijk, alleen maken we nu gebruik van een instructie om de invoer via de cursorbesturingstoetsen uit te lezen: STICK(0). In hoofdstuk 7 zult u zien dat deze instructie ook wordt gebruikt om de invoer via een joystick uit te lezen, alleen veranderen we dan de 0 in een andere waarde.

STICK(0) is een instructie die het adres uitleest waar de waarden van de cursorbesturingstoetsen worden opgeslagen op het moment dat een van deze toetsen wordt ingedrukt. De tekening geeft aan welke waarde in het betreffende adres wordt opgeslagen wanneer een toets wordt ingedrukt.



De waarden voor RECHTSBOVEN, RECHTSONDER, LINKSBOVEN en LINKSONDER verkrijgt u door twee toetsen tegelijk in te drukken. Drukt u geen toets in, dan wordt automatisch de waarde 0 opgeslagen. Het volgende programma is wat resultaat betreft gelijk aan het vorige, alleen wordt dat op een andere manier verkregen. Typ het in en RUN het.

```
10 REM BEWEGEN MET BEHULP VAN DE CURSORBESTUR
INGSTOETSEN
20 SCREEN 0
30 CLS
40 XA=20:XD=12
50 WA=XA:WD=XD
60 R=STICK(0)
70 IF R=1 THEN XD=XD-1
80 IF R=5 THEN XD=XD+1
```

```
90 IF R=7 THEN XA=XA-1
100 IF R=3 THEN XA=XA+1
110 IF XA<1 THEN XA=1
120 IF XA>39 THEN XA=39
130 IF XD<1 THEN XD=1
140 IF XD>23 THEN XD=23
150 VPOKE XA+XD*40,88
160 VPOKE WA+WD*40,32
170 GOTO 50
```

De kern van het programma is gelijk aan het vorige. Opnieuw hebben we enkele veranderingen aangebracht. De instructie A$=INKEY$ is vervangen door R=STICK(0). In deze regel wordt een waarde toegekend aan de variabele R (van richting). In de regels 70-100 worden deze waarden getest en afhankelijk van de waarde worden de variabelen XD en XA met 1 verhoogd of verlaagd. In dit programma wordt alleen gebruik gemaakt van de richtingen OMHOOG, NAAR RECHTS, OMLAAG en NAAR LINKS. U kunt het teken dus nog niet diagonaal over het scherm bewegen. Door gebruik te maken van deze methode kunt u een spelletje ontwerpen met een teken dat, gestuurd door de cursorbesturingstoetsen, in een vloeiende lijn over het scherm beweegt.

Met het volgende korte programmaatje controleert u de waarde van elke cursorbesturingstoets. Als u het RUNt, vliegt er een lange sliert, opgebouwd uit het getal 0, over het scherm. 0 is de waarde van het adres wanneer er geen toets wordt ingedrukt. Druk eens een cursortoets in en bekijk de veranderingen. Vergelijk de richting en het getal met de tekening op de vorige pagina en u zult zien dat deze overeenkomen. U kunt het programmaatje gebruiken om uit te vinden welke waarden u in een spel moet gebruiken om de richtingen te bepalen.

```
10 R=STICK(0)
20 PRINT R
30 GOTO 10
```

Hoofdstuk 6

We passen de stukjes in elkaar

In de afgelopen hoofdstukken zijn we heel wat wijzer geworden. We kunnen een teken uit zichzelf over het scherm laten stuiteren en met het toetsenbord of de cursorbesturingstoetsen de bewegingen van een karakter controleren. Maar we hebben nog steeds geen spel. Als we de skippy-bal en het bewegende teken combineren, gaan we de goede kant op. Het volgende programma is een eenvoudig en instructief spelletje. Het heeft geen overdreven versieringen. Muziek en grafiek ontbreken nog, evenals een titelpagina en een waardige afsluiting. Maar voor ons eerste programma is het resultaat heel acceptabel. Het is opzettelijk zo simpel gehouden, zodat duidelijk is hoe de skippy-bal en het bewegende teken tot dit spelletje hebben geleid. U zult routines herkennen die we al gebruikt en besproken hebben. Deze worden gecombineerd met een aantal nieuwe routines. Typ het programma nauwkeurig in en RUN het. Speel het spelletje een paar keer om te zien hoe het programma werkt. Daarna splitsen we het op in afzonderlijke delen en bespreken die. De besturing vindt plaats met behulp van de cursortoetsen.

```
10 REM SQUASH
20 GOSUB 300
30 GOSUB 420
40 PRINT CHR$(11):PRINT"SCORE: ";SC;" BALLEN:
 ";LL
50 VPOKE 6144+BX+BY*32,32
60 BX=BX+X
70 IF VPEEK(6144+BX+BY*32)=219 THEN GOSUB 240
80 IF VPEEK(6144+BX+BY*32)=22 THEN SC=SC+1:GO
SUB 240
```

```
90 BY=BY+Y
100 IF VPEEK(6144+BX+BY*32)=219 THEN GOSUB 27
0
110 IF VPEEK(6144+BX+BY*32)=22 THEN SC=SC+1:G
OSUB 240
120 IF BX<2 THEN GOSUB 520
130 VPOKE 6144+BX+BY*32,9
140 FOR TD=1 TO 15:NEXT TD
150 R=STICK(0)
160 IF R=1 THEN RY=RY-1
170 IF R=5 THEN RY=RY+1
180 IF RY<5 THEN RY=5
190 IF RY>17 THEN RY=17
200 VPOKE 6144+RX+RY*32,22
210 VPOKE 6144+RX+(RY-1)*32,32
220 VPOKE 6144+RX+(RY+1)*32,32
230 GOTO 40
240 X=-X
250 BX=BX+X
260 RETURN
270 Y=-Y
280 BY=BY+Y
290 RETURN
300 REM INITIALISEREN
310 K=8192
320 CLS
330 SCREEN 1
340 COLOR 15,4,11
350 BX=5:BY=5
360 RX=4:RY=12
370 X=1:Y=1
380 LL=3
390 VPOKE K+27,128
400 VPOKE K+1,20
410 RETURN
420 FOR SX=6277 TO 6293
430 VPOKE SX,219
440 NEXT SX
450 FOR SY=6725 TO 6741
460 VPOKE SY,219
470 NEXT SY
480 FOR SY=6325 TO 6741 STEP 32
490 VPOKE SY,219
500 NEXT SY
```

```
510 RETURN
520 IF LL=0 THEN GOTO 590
530 LL=LL-1
540 BX=5:BY=12
550 X=1
560 Y=1
570 FOR TD=1 TO 1000:NEXT TD
580 GOTO 40
590 FOR TD=1 TO 1000:NEXT TD
600 PRINT"HET SPEL IS AFGELOPEN"
610 END
```

De eerste programmaregel roept de subroutine aan die begint op regel 300.

```
300 REM INITIALISEREN
310 K=8192
320 CLS
330 SCREEN 1
340 COLOR 15,4,11
350 BX=5:BY=5
360 RX=4:RY=12
370 X=1:Y=1
380 LL=3
390 VPOKE K+27,128
400 VPOKE K+1,20
410 RETURN
```

Deze routine kent waarden toe aan de variabelen die de positie van de bal bepalen. Regel 360 doet hetzelfde voor de variabelen die de positie van het racket bepalen. Regel 370 initialiseert de variabelen X en Y. Deze worden gebruikt om de variabelen BX en BY te verhogen. Regel 380 bepaalt de variabele LL (aantal beurten). Op elk moment van het spel geeft deze variabele aan hoeveel beurten er nog zijn te gaan. Regel 410 brengt ons terug in het hoofdprogramma. Regel 30 roept de subroutine aan die begint op regel 420.

```
420 FOR SX=6277 TO 6293
430 VPOKE SX,219
440 NEXT SX
450 FOR SY=6725 TO 6741
460 VPOKE SY,219
470 NEXT SY
```

```
480 FOR SY=6325 TO 6741 STEP 32
490 VPOKE SY,219
500 NEXT SY
510 RETURN
```

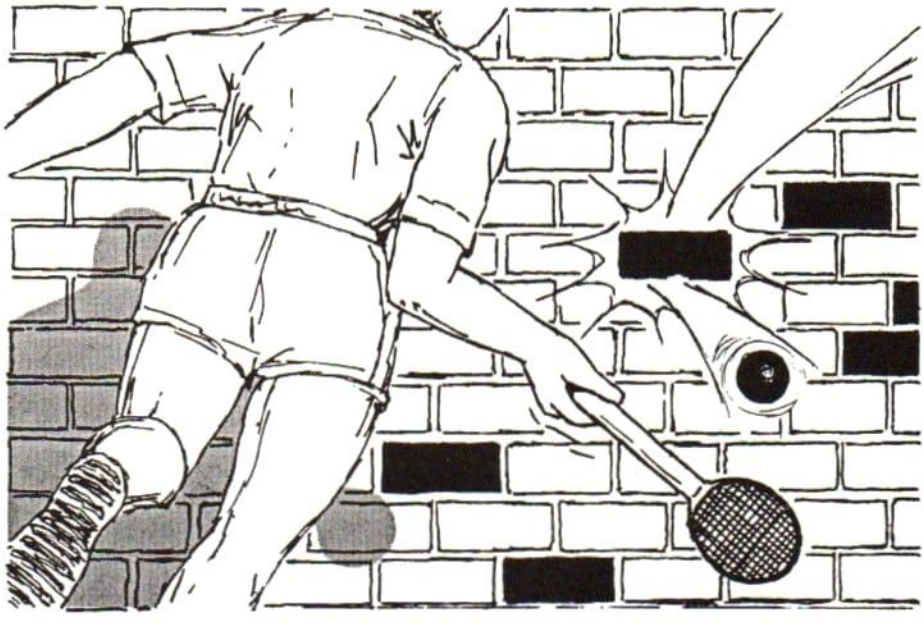
Deze routine gebruikt drie FOR/NEXT-lussen om met karakter 219 het speelveld op het scherm te POKEn. Nadat de drie zijden van het squashveld zijn verschenen, stuurt regel 510 ons terug naar het hoofdprogramma.

```
50 VPOKE 6144+BX+BY*32,32
60 BX=BX+X
70 IF VPEEK(6144+BX+BY*32)=219 THEN GOSUB 240
80 IF VPEEK(6144+BX+BY*32)=22 THEN SC=SC+1:GO
SUB 240
90 BY=BY+Y
100 IF VPEEK(6144+BX+BY*32)=219 THEN GOSUB 27
0
110 IF VPEEK(6144+BX+BY*32)=22 THEN SC=SC+1:G
OSUB 240
120 IF BX<2 THEN GOSUB 520
130 VPOKE 6144+BX+BY*32,9
140 FOR TD=1 TO 15:NEXT TD
150 R=STICK(0)
160 IF R=1 THEN RY=RY-1
170 IF R=5 THEN RY=RY+1
180 IF RY<5 THEN RY=5
190 IF RY>17 THEN RY=17
200 VPOKE 6144+RX+RY*32,22
210 VPOKE 6144+RX+(RY-1)*32,32
220 VPOKE 6144+RX+(RY+1)*32,32
230 GOTO 40
```

De spelroutine zelf zit in de lus tussen de regels 40 en 230. In wezen is deze lus opgebouwd uit de twee routines die we al kennen: de stuiterroutine en de lus die een teken over het scherm beweegt.

Scoren

Eerst PRINT regel 40 het woord SCORE, gevolgd door de waarde van variabele SCORE. Bij de start van het spel is de waarde van SC 0. Iedere keer dat het racket de bal raakt, wordt SC met 1 verhoogd. Verder wordt aangegeven hoeveel ballen de speler nog over heeft. Regel 50 VPOKEt een spatie op de plaats die de bal het laatst bezette. De horizontale positie van de bal wordt bijgehouden door regel 60. Vervolgens controleert het programma met VPEEK of de bal een van de randen van het speelveld heeft geraakt. Bij een botsing wordt de routine in regel 240 aangeroepen. Regel 80 controleert met VPEEK of de bal het racket heeft geraakt. Ons racket is een grafisch karakter (karakter 22, GRAPH +). Regel 90 en 100 zouden gemakkelijk kunnen worden gecombineerd. We hebben ze voor de duidelijkheid uit elkaar gehouden. Regel 90 houdt de verticale positie van de bal bij. Dan testen we in regel 100 of de bal botst met een wand ten gevolge van een verandering van de Y-positie. Het is nu niet nodig na te gaan of de bal het racket raakt. Is dat het geval, dan verandert alleen de X-richting. Of de waarde van BX kleiner is dan 2 wordt bekeken in regel 120. Zo ja, dan heeft de bal het racket gemist en is helemaal naar de linkerrand van het scherm gegaan. Als dit gebeurt, springen we met GOSUB naar regel 520. De bal komt overeen met karakter 9 (GRAPH + 0).



Een balletje slaan

In regel 150 begint de routine waarmee u het racket hanteert. De instructie STICK(0) bekijkt welke van de cursorbesturingstoetsen u hebt ingedrukt en kent de waarde ervan toe aan de variabele R. Drukt u op CURSOR OMHOOG dan krijgt R de waarde 1, bij CURSOR OMLAAG krijgt R de waarde 5. Elke andere richting wordt genegeerd. U zult opmerken dat alleen de Y-richting van het racket (variabele RY) wordt veranderd. De waarde van RX is al ingesteld in de initialisatieroutine en blijft ongewijzigd gedurende het spel. Dat betekent dat het racket alleen verticaal beweegt en niet horizontaal. Regel 180 en 190 voorkomen dat de bal de lijnen van het speelveld overschrijdt. Regel 200 VPOKEt het racket op het scherm en de volgende twee regels plaatsen een spatie aan weerszijden van de actuele positie van het racket. Dit om te voorkomen dat er meer dan één racket tegelijk op het scherm verschijnt. Regel 230 stuurt ons terug naar het begin van de lus. De twee routines in regel 240 en 270, die de waarden van X en Y veranderen blijven hetzelfde.

```
240 X=-X
250 BX=BX+X
260 RETURN
270 Y=-Y
280 BY=BY+Y
290 RETURN
```

We hebben een nieuwe routine toegevoegd, die begint in regel 520. Deze routine wordt opgeroepen als het racket de bal mist.

```
520 IF LL=0 THEN GOTO 590
530 LL=LL-1
540 BX=5:BY=12
550 X=1
560 Y=1
570 FOR TD=1 TO 1000:NEXT TD
580 GOTO 40
```

Regel 520 test eerst of er nog beurten over zijn. Als LL gelijk is aan 0, is het spel voorbij. Is dat niet het geval dan trekt deze routine 1 af van het aantal beurten (LL). Het programma gaat verder in regel 590, die voor een korte vertraging zorgt. Vervolgens PRINT regel 600 op het scherm de melding dat het spel afgelopen is. Als LL groter is dan 0, gaat het spel verder. De startpositie van de bal wordt opnieuw ingesteld en in de regels 550 en 560 worden de variabelen X en Y opnieuw geïnitialiseerd. Bij de start van het spel beweegt de bal schuin naar beneden in de richting van de achterzijde van het speelveld. Regel 570 zorgt voor een kleine vertraging en regel 580 brengt ons weer terug in het spel.

We hebben nu het programma Skippy-bal gecombineerd met een routine die een racket op het scherm plaats en ons in staat stelt dit via de cursorbesturingstoetsen te bewegen: het eerste spel met bewegende grafiek is een feit.

Hoofdstuk 7

De joystick

Met de cursortoetsen een figuurtje over het scherm bewegen is ook niet alles. Dat heeft u vast wel al gemerkt. Gelukkig kunnen op de MSX twee joysticks worden aangesloten. Onder het kopje 'Het alternatief' hebben we vermeld dat de instructie voor het uitlezen van waarden van de cursortoetsen gelijk is aan die van de joystick. Het is dan ook een fluitje van een cent om in een programma besturingstoetsen te vervangen door een joystick.

Voer het volgende programma in, plug een joystick in poort 1 aan de rechterkant van de computer en beweeg de X naar believen over het scherm. Links, rechts, naar boven, naar beneden, diagonaal - met een joystick gaat alles veel soepeler. Door de knop in te drukken wordt de boodschap VUUR!! op het scherm gePRINT.

```
10 REM VOORTDURENDE BEWEGING MET BEHULP VAN D
E JOYSTICK
20 CLS:REM SCHOON SCHERM
30 SCREEN 0
40 XA=20:XD=12
50 WA=XA:WD=XD
60 GOSUB 280
70 IF R=1 THEN XD=XD-1:REM OMHOOG
80 IF R=2 THEN XD=XD-1:XA=XA+1:REM OMHOOG REC
HTS
90 IF R=3 THEN XA=XA+1:REM RECHTS
100 IF R=4 THEN XD=XD+1:XA=XA+1:REM OMLAAG RE
CHTS
110 IF R=5 THEN XD=XD+1:REM OMLAAG
120 IF R=6 THEN XD=XD+1:XA=XA-1:REM OMlAAG LI
```

```
NKS
130 IF R=7 THEN XA=XA-1:REM LINKS
140 IF R=8 THEN XD=XD-1:XA=XA-1:REM OMHOOG LI
NKS
150 IF V<>0 THEN GOSUB 230
160 IF XA<0 THEN XA=0
170 IF XA>39 THEN XA=39
180 IF XD<0 THEN XD=0
190 IF XD>23 THEN XD=23
200 VPOKE WA+WD*40,32
210 VPOKE XA+XD*40,88
220 GOTO 50
230 REM VUUR ROUTINE
240 LOCATE 20,12:PRINT"VUUR!!"
250 FOR TD=1 TO 500:NEXT TD
260 LOCATE 20,12:PRINT"      "
270 RETURN
280 REM LEES DE JOYSTICK UIT
290 R=STICK(0)
300 V=STRIG(0)
310 RETURN
```

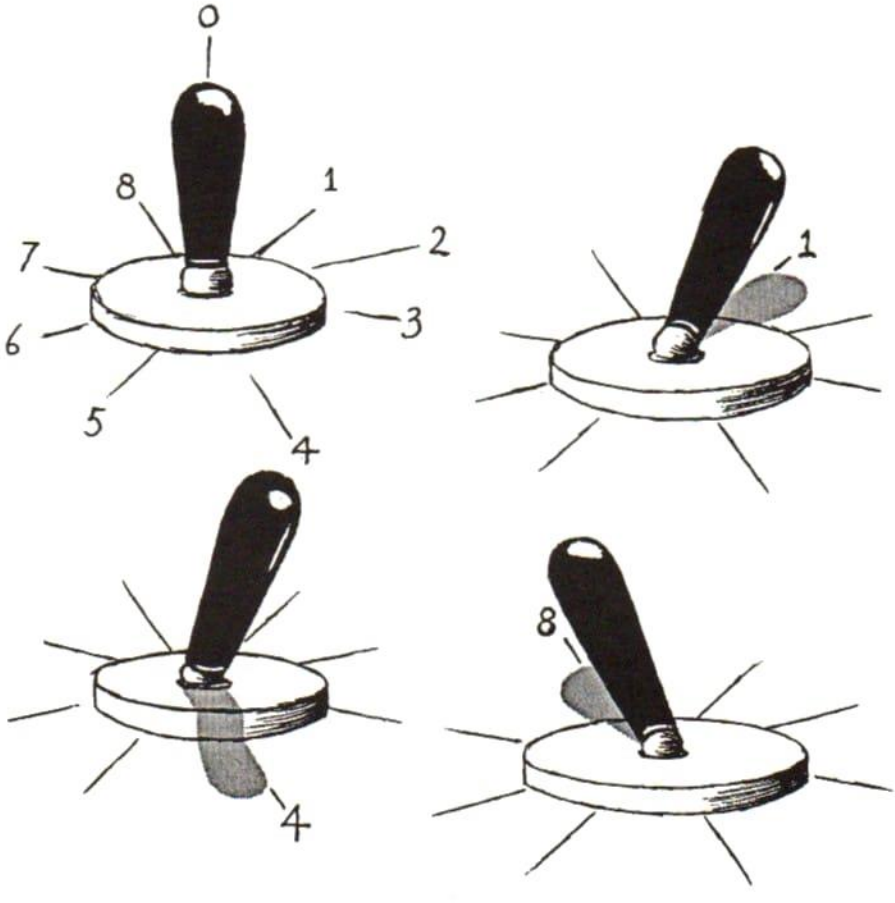
De routine die het teken over het scherm beweegt, is inmiddels bekend. Het enige verschil is dat de variabele R in de voorgaande programma's alleen een waarde voor omhoog, omlaag, rechts en links kon krijgen. De vier diagonale richtingen zijn nu ook gedefinieerd en daardoor u kunt het teken ook diagonaal over het scherm bewegen. De verandering die is aangebracht is dat STICK(n) nu niet waarde 0 maar waarde 1 krijgt wanneer u de joystick in poort 1 steekt en waarde 2 als hem in poort 2 steekt.

Om de vuurknop uit te lezen hebben we een extra instructie nodig: STRIG(n). N kan vijf verschillende waarden krijgen, lopend van 0 tot 4. Onderstaande tabel laat zien wat voor effect de verschillende waarden hebben.

waarde van n	**uitgelezen wordt**
0	**spatiebalk**
1	**joystick 1, vuurknop 1**
2	**joystick 2, vuurknop 1**
3	**joystick 1, vuurknop 2**
4	**joystick 2, vuurknop 2**

Hierbij moet wel worden opgemerkt dat de waarden 3 en 4 alleen werken als u over een joystick beschikt die is uitgevoerd volgens de MSX-standaard. Hebt u een willekeurige andere joystick, dan hebben STICK(3) en STICK (4) geen effect. De vuurknoppen van dergelijke joysticks worden niet afzonderlijk uitgelezen.

Terug naar het programa: regel 60 roept de subroutine op die in regel 280 begint. Deze routine kent waarden toe aan de R en de V, die respectievelijk de richting en het wel of niet indrukken van de vuurknop bepalen.



```
280 REM LEES DE JOYSTICK UIT
290 R=STICK(0)
300 V=STRIG(0)
310 RETURN
```

STICK(1) leest de joystick uit. We hebben n de waarde 1 gegeven omdat de joystick op poort 1 is aangesloten. STRIG(1) leest de vuurknop uit. Nadat de computer de subroutine in de regels 280-310 heeft beëindigd, keren we met RETURN terug naar het hoofdprogramma.

```
70 IF R=1 THEN XD=XD-1:REM OMHOOG
80 IF R=2 THEN XD=XD-1:XA=XA+1:REM OMHOOG REC
HTS
90 IF R=3 THEN XA=XA+1:REM RECHTS
100 IF R=4 THEN XD=XD+1:XA=XA+1:REM OMLAAG RE
CHTS
110 IF R=5 THEN XD=XD+1:REM OMLAAG
120 IF R=6 THEN XD=XD+1:XA=XA-1:REM OMlAAG LI
NKS
130 IF R=7 THEN XA=XA-1:REM LINKS
140 IF R=8 THEN XD=XD-1:XA=XA-1:REM OMHOOG LI
NKS
150 IF V<>0 THEN GOSUB 230
```

In de regels 70-140 handelt het programma overeenkomstig de waarde van R. De variabelen XA (horizontale positie) en XD (verticale positie) worden op de juiste manier gewijzigd. Regel 150 test variabele V om te zien of de vuurknop is ingedrukt. Als de waarde van V in regel 150 niet gelijk is aan 0, is de knop ingedrukt. De VUUR-routine, die begin in regel 230, wordt dan opgeroepen.

```
230 REM VUUR ROUTINE
240 LOCATE 20,12:PRINT"VUUR!!"
250 FOR TD=1 TO 500:NEXT TD
260 LOCATE 20,12:PRINT"      "
270 RETURN
```

Als het programma in deze regel terecht komt, PRINT regel 240 de melding VUUR op het scherm. We gebruiken een korte FOR/NEXT-lus (regel 250) om voor een vertraging te zorgen. Met een reeks spa-

ties wist regel 260 de melding. Met RETURN keren we weer terug naar het hoofdprogramma.

De VUUR-routine is maar een voorbeeld. U kunt hem natuurlijk vervangen door een willekeurige andere. Bijvoorbeeld door een routine die een raket lanceert, of door eentje die de snelheid van een ruimteschip of van een ander voertuig verhoogt of verlaagt. De mogelijkheden zijn legio.

Hoofdstuk 8

Strings

Op de MSX kunt u met strings hele aardige effecten bereiken. Eerst iets over het begrip 'string'. In de voorgaande programma's hebben we gezien dat we letters in plaats van getallen of waarden mochten gebruiken. De variabelen BX en BY bijvoorbeeld gebruikten we in het squash-programma om de horizontale en verticale positie van de bal op het scherm bij te houden. Zulke variabelen heten numerieke variabelen. Als een variabele moet bestaan uit letters of een combinatie van letters en cijfers, moeten we een stringvariabele gebruiken. Een stringvariabele is te herkennen aan het dollarteken ($) achter de naam van de variabele. Voorbeelden hiervan zijn A$ en B$ uit de programma's voor vloeiende beweging. Het dollarteken geeft aan dat de variabele letters of een combinatie van letters en andere tekens bevat. Bij stringvariabelen moeten de tekens tussen aanhalingstekens staan:

```
A$ = "COMPUTER"
```

In dit voorbeeld is het woord COMPUTER een string en staat tussen aanhalingstekens. Als u de instructie

```
PRINT A$
```

invoert, wordt het woord COMPUTER (de stringwaarde van A$) gePRINT. Voorbeelden hiervan vindt u in eerdere programma's, waar een stringvariabele werd gebruikt om de invoer van het toetsenbord in op te slaan. Als u een van deze programma's nog eens bekijkt, zult u zien dat de invoer via het toetsenbord tussen aanhalingstekens stond.

MSX-BASIC kent drie functies om trucjes met strings uit te halen. De eerste twee komen het meeste voor.

LEFT$	**(string links)**
RIGHT$	**(string rechts)**
MID$	**(string midden)**

We beginnen met LEFT$. Hiermee kunnen we een deel van een string afzonderlijk behandelen, te beginnen aan de linkerkant. Typ de volgende regels eens in en RUN ze.

```
10 REM DE INSTRUCTIE 'LEFT$' IN WERKING
20 CLS
30 A$="COMPUTER ANIMATIE"
40 B$=LEFT$(A$,3)
50 C$=LEFT$(A$,6)
60 D$=LEFT$(A$,9)
70 E$=LEFT$(A$,12)
80 F$=LEFT$(A$,15)
90 PRINT A$
100 PRINT B$
110 PRINT C$
120 PRINT D$
130 PRINT E$
140 PRINT F$
```

Op het scherm staat nu:

```
COMPUTER ANIMATIE
COM
COMPUT
COMPUTER
COMPUTER ANI
COMPUTER ANIMAT
```

Dit programma laat precies zien wat LEFT$ doet. U kunt vanaf de linkerkant van de string een bepaald aantal tekens overnemen. Let op de opbouw van de instructie. LEFT$ wordt gevolgd door de variabelenaam van de string die u wilt gebruiken, een komma en tot slot het aantal tekens dat u wilt overnemen. De variabelenaam en het aantal staan tussen haakjes.

RIGHT$ functioneert op dezelfde manier, maar dan vanaf de rechter-

kant van de string. Het volgende programma laat de werking van RIGHT$ zien:

```
10 REM DE INSTRUCTIE 'RIGHT$' IN WERKING
20 CLS
30 A$="COMPUTER ANIMATIE"
40 B$=RIGHT$(A$,3)
50 C$=RIGHT$(A$,6)
60 D$=RIGHT$(A$,8)
70 E$=RIGHT$(A$,12)
80 F$=RIGHT$(A$,15)
90 PRINT A$
100 PRINT B$
110 PRINT C$
120 PRINT D$
130 PRINT E$
140 PRINT F$
```

Na RUN verschijnt op het scherm:

```
COMPUTER ANIMATIE
TIE
IMATIE
ANIMATIE
TER ANIMATIE
MPUTER ANIMATIE
```

MID$ werkt iets anders. Met deze instructie kunt u midden in een string vanaf een bepaalde positie een aantal tekens overnemen. Het volgende programma toont MID$ in actie.

```
10 REM DE INSTRUCTIE 'MID$' IN WERKING
20 CLS
30 A$="COMPUTER ANIMATIE"
40 B$=MID$(A$,2,5)
50 C$=MID$(A$,3,3)
60 D$=MID$(A$,5,8)
70 E$=MID$(A$,4,10)
80 F$=MID$(A$,12,3)
90 G$=MID$(A$,8,4)
```

```
100 PRINT A$
110 PRINT B$
120 PRINT C$
130 PRINT D$
140 PRINT E$
150 PRINT F$
160 PRINT G$
```

Op het scherm is het volgende te zien:

```
COMPUTER ANIMATIE
OMPUT
MPU
UTER ANI
PUTER ANIM
IMA
R AN
```

Bij de MID$-instructie geeft u de string aan waarmee u wilt werken, gevolgd door het nummer van het teken waarmee u wilt beginnen en het aantal tekens dat u wilt overnemen.

Strings aan elkaar rijgen

Strings kunnen ook bij elkaar worden opgeteld om nieuwe strings te vormen. Een moeilijk woord hiervoor is *concatenatie*. Een klein programma demonstreert het principe:

```
10 REM HET OPTELLEN VAN REEKSEN
20 CLS
30 A$="COMPUTER"
40 B$="ANIMATIE"
50 C$="AANEENSCHAKELING"
60 D$=A$+C$+B$
70 PRINT D$
```

Na RUN verschijnt het afschuwelijke woord

```
COMPUTERAANEENSCHAKELINGANIMATIE
```

Met LEFT$, RIGHT$ en MID$ erbij kunnen we een hoop lol beleven:

```
10 REM HET OPTELLEN VAN STRINGS
20 CLS
30 A$="COMPUTER"
40 B$="TOETSENBORD"
50 C$="ALFABETISCH"
60 D$=A$+C$+B$
70 X=INT(RND(1)*3)+1
80 Y=INT(RND(1)*5)+1
90 E$=MID$(D$,X+3,Y+5)
100 F$=MID$(C$,Y,X)
110 G$=RIGHT$(B$,Y)
120 H$=LEFT$(A$,Y)
130 PRINT E$+H$
140 PRINT G$+H$+F$
150 PRINT F$+E$
160 PRINT E$+G$
170 PRINT H$+E$
180 PRINT F$+H$
```

```
UTERALC
DCAL
ALUTERAL
UTERALD
CUTERAL
ALC
```

```
TERALFABCOM
ORDCOMFAB
FABTERALFAB
TERALFABORD
COMTERALFAB
FABCOM
```

```
TERALFC
DCALF
ALFTERALF
TERALFD
CTERALF
ALFC
```

Onder de listing staan enkele voorbeelden van de uitvoer van dit programma.

We kunnen rare effecten bereiken door gebruik te maken van deze stringfuncties. Dit soort effecten kan worden gebruikt voor een titelpagina of om een programma op te sieren. Met MID$ kunnen we een reeks tekens een voor een op het scherm laten verschijnen en in het midden laten stoppen. Dat kan met een heel kort programma:

```
10 REM TITEL BEWEGEN MET BEHULP VAN DE INSTRU
CTIE MID$
20 CLS:KEY OFF
30 X$="          DIT IS EEN BEWEGENDE TITEL":R
EM 6 SPATIES
40 TX=40
50 FOR Y=1 TO 36
60 X2$=MID$(X$,TX,Y)
70 LOCATE 0,10:PRINT X2$
80 GOSUB 120
90 TX=TX-1
100 NEXT Y
110 GOTO 110
120 FOR TD=1 TO 50:NEXT TD
130 RETURN
```

Regel 30 produceert de string waar we mee gaan werken. Let erop dat als er spaties in een string aanwezig zijn, deze als tekens in die string worden meegeteld. Om de string in het midden van het scherm te laten stoppen moet u voldoende spaties in de string opnemen. De string DEZE TITEL BEWEEGT bestaat uit 18 tekens, de spaties meegeteld. De default breedte van het scherm in SCREEN 0 is 37 dus moeten er, om deze string precies midden op het scherm te krijgen, 10 spaties aan voorafgaan.

Goochelen

Regel 40 bepaalt de waarde van variabele TX. Die is gelijk aan het totale aantal posities in de string. Regel 50 is het begin van een FOR/NEXT-lus die tot datzelfde aantal telt. MID$ (regel 60) gebruiken we om met de tekens te goochelen. Deze regel neemt een deel van de string X$, te beginnen op de positie gelijk aan de waarde van TX. Het

aantal tekens dat wordt overgenomen, is gelijk aan de waarde van Y. Het resultaat wordt toegekend aan variabele X2$. De eerste keer dat de lus wordt doorlopen, neemt regel 60 het laatste teken van X$. Bij de volgende doorloop is TX met 1 verlaagd en de waarde van Y met 1 verhoogd. Regel 60 neemt dan de laatste twee tekens van X$ en PRINT deze op het scherm. Dit gaat zo door tot de hele FOR/NEXT-lus is doorlopen. Regel 70 gebruikt een LOCATE-instructie om er zeker van te zijn dat X2$ altijd op dezelfde regel wordt gePRINT. De LOCATE-instructie in deze PRINT-opdracht is van wezenlijk belang. Laat u die maar eens weg, RUN het programma opnieuw en bekijk het resultaat.

Door een combinatie van MID$, LEFT$ en RIGHT$ te gebruiken, kunnen we een titel produceren die onafgebroken van rechts naar links over het scherm beweegt. De lus in het volgende programma wordt eindeloos herhaald. Met CTRL-STOP kunt u het programma onderbreken.

```
10 REM TITEL BEWEGEN 2
20 CLS:KEY OFF
30 X$="DIT IS EEN BEWEGENDE TITEL "
40 X$=X$+X$
50 Y$=RIGHT$(X$,LEN(X$)-1)+LEFT$(X$,1)
60 X$=Y$
70 Y$=MID$(Y$,1,37)
80 LOCATE 0,10:PRINT Y$
90 FOR TD=1 TO 30:NEXT TD
100 GOTO 50
```

Deze routine is een verbetering voor elk programma. In regel 30 staat de string X$. Die bevat de tekst die we op het scherm willen zetten. Deze routine vereist een tekst van tenminste 40 tekens om goed te werken. Als de string minder dan 40 tekens bevat, tellen we hem bij zichzelf op om de string voldoende lengte te geven. Dit gebeurt in regel 40. Regel 50 is heel belangrijk. Hier wordt een nieuwe functie geïntroduceerd: LEN. LEN geeft het aantal tekens in de string, in dit geval X$. Simpel gezegd pakt regel 50 het linker teken van X$ en voegt het rechts toe. In regel 60 wordt deze nieuwe string toegekend aan X$. Regel 70 controleert met MID$ of Y$ de eerste 40 tekens van de string bevat. Wanneer Y$ wordt gePRINT in regel 80, past dit mooi op het scherm.

Op het scherm

In de vorige programma's bent u LOCATE weer meerdere malen tegengekomen. In hoofdstuk 1 hebben we deze instructie al nader bekeken. Toch komen we er nog even op terug. We geven u een kort programma dat met behulp van LOCATE en X- en Y-variabelen strings op willekeurige plaatsen op het scherm zet.

```
10 REM LOCATE ALLES OP HET SCHERM
20 CLS
30 KEY OFF
40 Q=RND(-TIME)
50 X$="MSX"
60 Y$="COMPUTER"
70 FOR A=1 TO 20
80 X=INT(RND(1)*28)+1
90 Y=INT(RND(1)*23)
100 GOSUB 130
110 NEXT A
120 END
130 LOCATE X,Y
140 IF F=1 THEN PRINT X$:ELSE PRINT Y$
150 IF F=1 THEN F=0 ELSE F=1
160 FOR TD=1 TO 250:NEXT TD
170 RETURN
```

Nadat we de in de regels 80 en 90 de variabelen X en Y een willekeurige waarde hebben gegeven, springen we middels regel 100 naar de subroutine in de regels 130-170. In de subroutine wordt de plaats op het scherm bepaald met behulp van LOCATE. Daarna wordt er bekeken welke van de twee strings (X$ of Y$) moet worden gePRINT. Regel 160 zorgt voor een kleine vertraging en de RETURN in regel 170 stuurt het programma weer terug, waarna de volgende X- en Y-positie kunnen worden bepaald.

Hoofdstuk 9

Volop actie

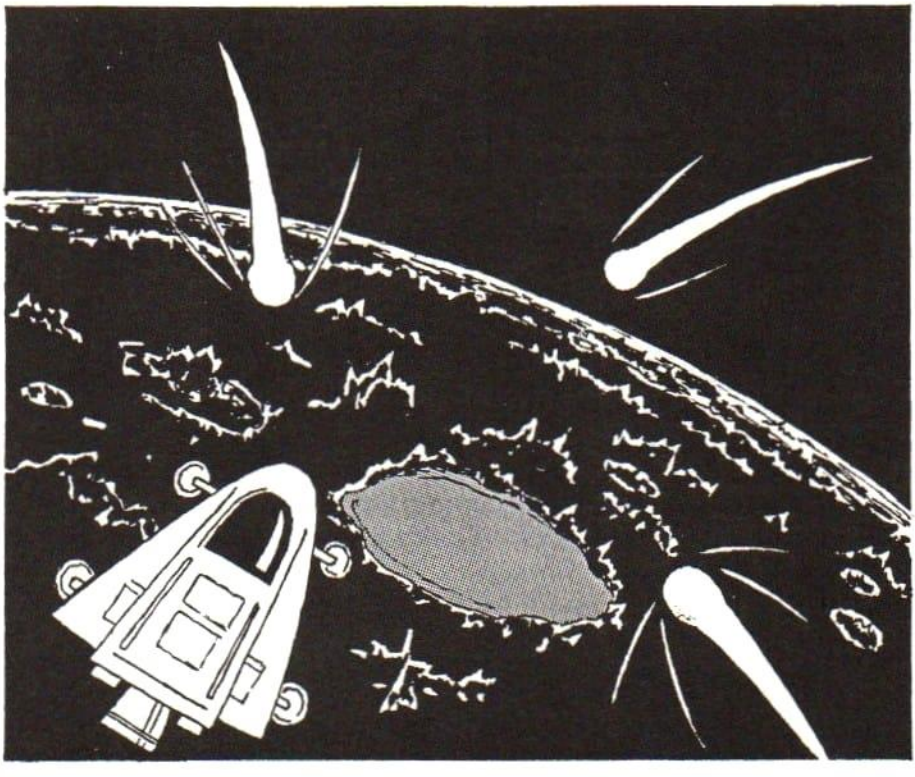
Laten we onze pas verworven grafische vaardigheden eens in praktijk brengen en een actiespel maken. Om een spelletje te schrijven moeten we in ieder geval een idee hebben van de omvang en het speltype. In de loop van het programmeerproces wordt er meestal zo geschaafd en gesleuteld aan het oorspronkelijke raamwerk dat dat in het eindprodukt nauwelijks meer te herkennen is. Dat is een van de avontuurlijke kanten van eigen programma's. Een spelletje is nooit echt af. Er valt altijd wel wat nieuws te verzinnen of te perfectioneren.

Voor ons eerste complete spel hebben we hopelijk niet te hoog gegrepen: we gaan landen op de maan. Er zijn verschillende variaties op dit thema, maar het type spel is heel geschikt om mee te beginnen. Het bevat alle eigenschappen die we tot nu toe hebben gebruikt, aangevuld met een paar nieuwe klankeffecten. De bedoeling is om uw ruimteschip veilig op het maanoppervlak te laten landen. Allereerst hebben we een ruimteschip nodig, dat van boven naar beneden over het scherm kan bewegen. Daarnaast moeten we de beweging van het schip helemaal in de hand hebben. Ten derde is een veilig landingsgebied onderaan het scherm onontbeerlijk. Verder moeten we kunnen controleren of het vaartuig veilig in dit gebied is geland of dat het is neergestort op het maanoppervlak. Een wezenlijk onderdeel van de meeste spelletjes is het scorebord. Records zijn er om te worden gebroken en vormen een uitdaging voor elke speler.

Complicaties

Zomaar op de maan landen is natuurlijk geen kunst. Een actiespel moet avontuurlijk zijn, met andere woorden: er moeten bepaalde

moeilijkheden worden overwonnen. In dit programma, dat we MAANLANDING hebben genoemd, neemt na elke veilige landing de snelheid toe waarmee het ruimtevaartuig daalt. Tevens wordt het aantal obstakels na elke landing groter. Om het de speler nog lastiger te maken, begint de landing steeds op een andere positie. Tenslotte wordt het veilige landingsgebied ook nog eens verplaatst. Als we de basiselementen hebben ingevoerd en zeker weten dat ze onder alle omstandigheden op de juiste manier functioneren, kunnen we klank en heldere kleur toevoegen om het programma te verfraaien. De complete programmalisting van MAANLANDING volgt hieronder. Nadat u die hebt ingetypt en het spel enkele keren heeft gespeeld, splitsen we het programma op en bespreken de afzonderlijke routines.



```
10 REM MAANLANDING
20 GOSUB 560:REM INITIALISEREN
30 GOSUB 900:REM TITELSCHERM
40 GOSUB 720:REM SCHERM KLAARMAKEN
50 REM HOOFDLUS
60 VPOKE 6144+XX+XY*32,32
70 R=STICK(0)
80 IF R=7 THEN XX=XX-1
90 IF R=3 THEN XX=XX+1
100 XY=XY+1
110 GOSUB 660
120 IF VPEEK(6144+XX+XY*32)=42 THEN 180:REM A
STERIODE GERAAKT
130 IF VPEEK(6144+XX+XY*32)=215 THEN 430:REM
VEILIGE LANDING
140 IF XY>23 THEN 180:REM BASIS GEMIST
150 VPOKE 6144+XX+XY*32,6
160 FOR TF=1 TO TT:NEXT TF
170 GOTO 50
180 REM BOTSING
190 GOSUB 270
200 LL=LL-1
210 IF LL=0 THEN GOTO 1020
220 XY=1
230 XX=INT(RND(1)*20)+5
240 GOSUB 340
250 GOSUB 720
260 GOTO 50
270 SOUND 3,100
280 SOUND 6,20
290 SOUND 7,47
300 SOUND 9,9
310 FOR TD=1 TO 200:NEXT TD
320 FOR I=0 TO 9:SOUND I,0:NEXT I
330 RETURN
340 SOUND 0,100
350 SOUND 3,30
360 SOUND 6,15
370 SOUND 7,46
380 SOUND 8,9
390 SOUND 9,9
400 FOR TD=1 TO 100:NEXT TD
410 FOR I=0 TO 9:SOUND I,0:NEXT I
420 RETURN
```

```
430 REM VEILIGE LANDING
440 VPOKE 6816+XX,6
450 RX=XX
460 XX=INT(RND(1)*20)+5:XY=1
470 SC=SC+(1000-TT)*4
480 TT=TT-25
490 IF TT<0 THEN TT=0
500 FOR TD=1 TO 500:NEXT TD
510 VPOKE 6816+RX,32
520 NS=NS+5
530 GOSUB 340
540 GOSUB 720
550 GOTO 50
560 REM INITIALISEREN
570 XX=INT(RND(-TIME)*20)+10:XY=1
580 SCREEN 1
590 KEY OFF
600 K=8192
610 TT=300
620 NS=15
630 LL=4
640 COLOR 15,1,11
650 RETURN
660 SOUND 0,255
670 SOUND 7,62
680 SOUND 8,9
690 FOR TD=1 TO 10:NEXT TD
700 SOUND 8,0
710 RETURN
720 REM MAAK HET SCHERM KLAAR
730 CLS:REM SCHERM SCHOON
740 VPOKE K+3,49
750 FOR LZ=6848 TO 6878 STEP 2
760 VPOKE LZ,29
770 VPOKE LZ+1,30
780 NEXT LZ
790 LB=INT(RND(1)*20)+10
800 VPOKE 6848+LB,215:VPOKE 6848+LB+1,215:VPO
KE 6848+LB+2,215
810 VPOKE K,145
820 VPOKE K+5,177
830 FOR PS=1 TO NS
840 SP=INT(RND(1)*576)
850 VPOKE 6272+SP,42
```

```
860 NEXT PS
870 LOCATE 8,0:PRINT"MAANLANDING"
880 FOR TD=1 TO 300:NEXT TD
890 RETURN
900 REM TITELSCHERM
910 CLS:REM SCHOON SCHERM
920 LOCATE 8,11:PRINT"MAANLANDING"
930 REM GELUID
940 FOR Q=1 TO 255 STEP.5
950 SOUND 0,Q
960 SOUND 7,62
970 SOUND 8,9
980 NEXT Q
990 FOR Q=1 TO 8:SOUND Q,0:NEXT Q
1000 FOR TD=1 TO 1000:NEXT TD
1010 RETURN
1020 REM EINDSCORE
1030 CLS
1040 COLOR 15,4,4
1050 FOR TD=1 TO 2000:NEXT TD
1060 LOCATE 3,6:PRINT"HET SPEL IS AFGELOPEN"
1070 LOCATE 3,8:PRINT"UW EINDSCORE IS:";SC
1080 FOR TD=1 TO 2000:NEXT TD
1090 LOCATE 3,10:PRINT"WILT U OPNIEUW SPELEN"
1100 A$=INKEY$
1110 IF A$="J" THEN RUN
1120 IF A$<>"N" THEN GOTO 1100
1130 CLS
1140 PRINT"TOT ZIENS":END
```

Eerst moeten we enkele variabelen invoeren om het ruimteschip te kunnen bewegen. Ook hebben we variabelen nodig voor de vertragingslus die de snelheid van het ruimteschip regelt. Laten we de initialisatieroutine eens nader bekijken.

```
560 REM INITIALISEREN
570 XX=INT(RND(-TIME)*20)+10:XY=1
580 SCREEN 1
590 KEY OFF
600 K=8192
610 TT=300
620 NS=15
```

```
630 LL=4
640 COLOR 15,1,11
650 RETURN
```

De GOSUB-instructie (regel 20) roept de initialisatieroutine aan. Eerst voeren we variabelen in die de X- en Y-positie van het ruimteschip bijhouden (regel 570). Dat zijn de variabele XX en XY. U zult opmerken dat XX willekeurig wordt bepaald, terwijl XY de waarde 1 krijgt. Dit heeft als gevolg dat het ruimteschip op een willekeurige plaats vanaf de linkerzijde op het scherm begint, maar één regel vanaf de bovenkant. Om voor XX een echt willekeurig getal te genereren hebben we gebruik gemaakt van de mogelijkheid een random getal van -TIME te nemen. Neemt u een random waarde van 1, dan blijkt vrij snel dat random eigenlijk niet echt random is. PRINT u vijf maal achtereen een random waarde van 1, dan ziet u de volgende uitkomst:

```
.59521943994623
.10658628050158
.76597651772823
.57756392935958
.73474759503023
```

Het resultaat is voorspelbaar, want de MSX gebruikt het voorgaande random getal om het volgende te genereren.

In regel 580 wordt SCREEN 1 gekozen om gebruik te kunnen maken van de mogelijkheid om met kleur te spelen. Daartoe wordt in regel 600 de variabele K gedefinieerd. Regel 610 initialiseert de variabele TT. Deze variabele wordt gebruikt in een FOR/NEXT-lus om de beweging van het ruimteschip te vertragen. Na elke succesvolle landing wordt TT verlaagd, zodat het schip zich op de volgende vlucht sneller voortbeweegt. Regel 620 voert de variabele NS in. Deze variabele houdt bij hoeveel sterren er aan het firmament moeten verschijnen. Tijdens de eerste landingspoging zijn dat er 15, bij elke volgende landing worden het er meer. Nadat in regel 640 de gewenste kleur van voorgrond, achtergrond en beelschermrand is ingesteld, brengt regel 650 ons vervolgens terug naar het hoofdprogramma.

De volgende routine is de hoofdlus van het spel. Eerst moeten we ervoor zorgen dat het ruimteschip op de juiste wijze over het scherm beweegt. Dan volgt een test: is het ruimteschip veilig geland of heeft het het maanoppervlak geraakt? Daarna wordt gecontroleerd of het ruimteschip tegen een ster is gebotst. De kern van de hoofdlus

waarin beweging en besturing van het ruimteschip wordt geregeld, is het eerst geschreven. Vervolgens is de landingszone toegevoegd, zodat we kunnen nagaan of ons schip veilig is geland. We bekijken nu de hoofdlus gedetailleerd en gaan vervolgens naar de routine die het scherm opmaakt.

```
50 REM HOOFDLUS
60 VPOKE 6144+XX+XY*32,32
70 R=STICK(0)
80 IF R=7 THEN XX=XX-1
90 IF R=3 THEN XX=XX+1
100 XY=XY+1
110 GOSUB 660
120 IF VPEEK(6144+XX+XY*32)=42 THEN 180:REM A
STERIODE GERAAKT
130 IF VPEEK(6144+XX+XY*32)=215 THEN 430:REM
VEILIGE LANDING
140 IF XY>23 THEN 180:REM BASIS GEMIST
150 VPOKE 6144+XX+XY*32,6
160 FOR TF=1 TO TT:NEXT TF
170 GOTO 50
```

Zoals u ziet is de hoofdlus niet erg lang. Hij bevat niet veel meer dan de elementaire routine waar we al mee gewerkt hebben. Zo'n basisroutine wordt een spel door de manier waarop andere routines en effecten erin zijn verwerkt. In regel 60 staat de inmiddels bekende formule om een spatie op de actuele positie van het ruimteschip te VPOKEn. Daardoor bevindt zich steeds maar één ruimteschip op het scherm.

R=STICK(1)

in regel 70 leest de invoer van de joystick in poort 1. Bent u niet in het bezit van een joystick, dan kunt u de 1 vervangen door een 0. Dan wordt de invoer van de cursorbesturingstoetsen gelezen in plaats van de joystick. Duwt u de stick naar rechts (R=3) dan wordt XX met 1 verhoogd, bij een beweging naar links wordt XX met 1 verlaagd. Het ruimteschip beweegt zich naar rechts of naar links. De verhoging van XY met 1 zorgt voor de afdaling van het ruimteschip. Daarna wordt de subroutine in regel 660 aangeroepen die voor het geluid van de afdaling zorgt. Na afloop keren we terug naar de hoofdlus.

Botsingen opsporen

Met VPEEK sporen we in de regels 120 en 130 botsingen op. De nieuwe positie, bepaald door de waarden XX en XY, wordt getest voordat het ruimteschip werkelijk op het scherm wordt geVPOKEt. Regel 120 controleert of een ster wordt geraakt. Waarde 42 is de code voor de asterisk. Bij een botsing gaat het programma naar regel 180, waar een nieuwe routine begint. Regel 130 test of het ruimteschip zich boven het landingsgebied bevindt (waarde 215). Is dat geval, dan wordt de routine opgeroepen die voor een veilige landing zorgt. Regel 140 test of het ruimteschip het landingsgebied heeft gemist. VPEEK is daarbij niet nodig. Als de waarde van XY groter is dan 23, is de onderkant van het scherm bereikt en dus het landingsgebied gemist. In dat geval wordt de botsroutine aangesproken. Regel 150 VPOKEt het ruimteschip op het scherm. De vertragingslus in regel 160 regelt de snelheid waarmee het ruimteschip daalt. De doorlooptijd van de lus wordt kleiner als variabele TT wordt verlaagt. Regel 170 stuurt het programma terug naar het begin van de lus.

De schermopmaakroutine, belangrijk voor de opbouw van het programma, zet zowel het landingsgebied als de obstakels op het scherm. Een gedeelte van de routine hangt samen met de hoofdlus en wordt na elke afdaling opgeroepen om het plaatje weer op te bouwen. Er staan een aantal random variabelen in en een aantal variabelen die worden verhoogd als het spel verder gaat. Laten we de betreffende regels eens nauwkeurig bekijken.

```
720 REM MAAK HET SCHERM KLAAR
730 CLS:REM SCHERM SCHOON
740 VPOKE K+3,49
750 FOR LZ=6848 TO 6878 STEP 2
760 VPOKE LZ,29
770 VPOKE LZ+1,30
780 NEXT LZ
790 LB=INT(RND(1)*20)+10
800 VPOKE 6848+LB,215:VPOKE 6848+LB+1,215:VPO
KE 6848+LB+2,215
810 VPOKE K,145
820 VPOKE K+5,177
830 FOR PS=1 TO NS
840 SP=INT(RND(1)*576)
850 VPOKE 6272+SP,42
860 NEXT PS
```

```
870 LOCATE 8,0:PRINT"MAANLANDING"
880 FOR TD=1 TO 300:NEXT TD
890 RETURN
```

Aan het begin van deze routine wissen we het scherm, ongeacht wat daar nog op staat. Dat kan zowel de titel als het scherm van de vorige beurt zijn. In de regels 750-780 wordt de landingszone (LZ) op het scherm geVPOKEt met een nieuwe FOR/NEXT-lus. FOR/NEXT-lussen kunnen allerlei soorten herhalingsopdrachten uitvoeren. Deze lus wordt gevolgd door de instructie STEP 2. De lus begint met het eerste getal en telt daar bij elke doorloop 2 bij op. Dit deel wordt gebruikt om het teken met de waarde 29 op het scherm te VPOKEn. Regel 770 verhoogt de waarde van LZ met 1 en VPOKEt vervolgens het teken met de waarde 30 op het scherm. Samen vormen deze twee tekens de bergketen op de achtergrond. De regel voorafgaand aan deze lus (740) bepaalt de kleur van de tekens. In regel 790 wordt variabele LB (landingsbasis) gelijkgesteld aan een toevalswaarde tussen 10 en 30. Regel 800 VPOKEt waarde 215 (het hokjespatroon) op het scherm. De rest van de regel VPOKEt de waarde in de volgende twee schermposities om het landingsgebied te vormen. Regel 810 bepaalt de kleur van het ruimteschip en regel 820 de kleur van de landingsbasis. In regel 830-850 staat weer een FOR/NEXT-lus. Deze VPOKEt de sterren op het scherm. Variabele NS in regel 830 geeft het aantal sterren aan dat tijdens elke doorloop op het scherm moet worden geVPOKEt. NS wordt na elke veilige landing verhoogd. De positie van de ster (SP) wordt at random gekozen. Regel 840 genereert een getal tussen 0 en 576. Dit getal wordt bij 6272 opgeteld om de ster op het scherm te zetten. 6272 is de basiswaarde van het schermgeheugen in SCREEN 1, plus 5 x 32 om ervoor te zorgen dat er geen sterren op de eerste vijf regels en de laatste regel van het scherm verschijnen. Zo kan een ster zich niet in de titel of het landingsgebied bevinden. We hebben ook de eerste regels leeg gelaten. Als u het spel een tijdje speelt en het schip daalt bijna met de maximumsnelheid, hebt u weinig kans als het schip direct boven een ster begint. Een kleine adempauze is zeker nodig om de speler een eerlijke kans te geven.

Regel 870 PRINT de titel bovenaan het scherm. Hierbij wordt gebruik gemaakt van de LOCATE-instructie.

Landen en botsen

Regel 880 zorgt voor een korte pauze, zodat de speler zich gereed kan maken voor de volgende ronde. Het is belangrijk om zulke pauzes in

te lassen. Ze hoeven niet erg lang te zijn, maar ze moeten de speler de gelegenheid bieden te zien waar de landingsbasis en de obstakels zich bevinden, voordat het schip verschijnt. Regel 890 brengt ons terug naar de plaats waar we vandaan kwamen. Dat kan zowel de routine voor een veilige landing als de botsroutine zijn. Beide roepen de schermopmaakroutine aan voordat ze terugkeren naar het hoofdprogramma. De volgende routine zorgt voor de veilige landing.

```
430 REM VEILIGE LANDING
440 VPOKE 6816+XX,6
450 RX=XX
460 XX=INT(RND(1)*20)+5:XY=1
470 SC=SC+(1000-TT)*4
480 TT=TT-25
490 IF TT<0 THEN TT=0
500 FOR TD=1 TO 500:NEXT TD
510 VPOKE 6816+RX,32
520 NS=NS+5
530 GOSUB 340
540 GOSUB 720
550 GOTO 50
```

De routine moet ook variabelen opnieuw instellen die nodig zijn bij de volgende beurt. Het is belangrijk om goed bij u zelf na te gaan wat u van zo'n routine verwacht. Als u vergeet variabelen te herstellen en de betreffende waarden te verhogen, kunnen verwarring en problemen uw deel zijn. Regel 440 VPOKEt het ruimteschip op het landingsplatform. Het blijft op het scherm tijdens de uitvoering van dit deel van het programma. In regel 1030 staat een nieuwe variabele: RX. Deze variabele wordt gebruikt om de actuele waarde van XX te onthouden, zodat we er later een spatie overheen kunnen VPOKEn. Regel 460 initialiseert XX en XY opnieuw om de volgende landing voor te bereiden. De score wordt verhoogd in regel 470. Het scoresysteem is eenvoudig. De actuele waarde van TT wordt van 1000 afgetrokken en het resultaat wordt met 4 vermenigvuldigd. Hoe sneller het vaartuig daalt, des te hoger wordt de score. Als u wilt, kunt u door veranderingen in deze regel aan te brengen zelf een scoresysteem maken of het onze veranderen. We verhogen vervolgens de waarde van TT. Bedenk dat we eerst de score hebben uitgerekend. Het is belangrijk deze onderdelen in de juiste volgorde te behandelen. Regel 490 is een beveiliging. Deze regel is toegevoegd om er zeker van te zijn dat er geen problemen optreden wanneer de waarde van TT kleiner wordt dan 0. Regel 500 is een korte vertraging, waardoor het schip

op het landingsplatform zichtbaar is. Regel 510 VPOKEt met RX een spatie op het scherm om het schip te verwijderen. Regel 520 verhoogt de waarde van NS met 5. NS is het aantal sterren op het scherm. Regel 530 roept de subroutine op die voor het korte deuntje tussen twee beurten zorgt. De routine die in regel 540 wordt opgeroepen, herstelt het scherm. Regel 550 stuurt het programma terug naar het begin van de hoofdlus.

Dan volgt de botsroutine. Deze wordt opgeroepen als het schip in botsing komt met een ster of het landingsplatform mist:

```
180 REM BOTSING
190 GOSUB 270
200 LL=LL-1
210 IF LL=0 THEN GOTO 1020
220 XY=1
230 XX=INT(RND(1)*20)+5
240 GOSUB 340
250 GOSUB 720
260 GOTO 50
```

In regel 190 wordt de geluidsroutine opgeroepen die de klap van een botsing produceert. Regel 200 trekt 1 af van het resterende aantal beurten. Als die op zijn, wordt de eindroutine aangeroepen. Regel 220 en 230 herstellen de waarden van XX en XY zodat het schip weer bovenaan het scherm kan beginnen. Daarna worden de geluidsroutine en de schermopmaakroutine opgeroepen. Tot slot keert het programma terug naar het begin van de hoofdlus.

De finishing touch

De volgende routine is een waardige afsluiting. Op het scherm staat dat het spel voorbij is en hoe hoog uw score is. De beslissing om verder te spelen is aan u. Het programma is zo beleefd er even naar te vragen.

```
1020 REM EINDSCORE
1030 CLS
1040 COLOR 15,4,4
1050 FOR TD=1 TO 2000:NEXT TD
1060 LOCATE 3,6:PRINT"HET SPEL IS AFGELOPEN"
1070 LOCATE 3,8:PRINT"UW EINDSCORE IS:";SC
```

```
1080 FOR TD=1 TO 2000:NEXT TD
1090 LOCATE 3,10:PRINT"WILT U OPNIEUW SPELEN"
1100 A$=INKEY$
1110 IF A$="J" THEN RUN
1120 IF A$<>"N" THEN GOTO 1100
1130 CLS
1140 PRINT"TOT ZIENS":END
```

In regel 1030 wordt het scherm gewist. Het scherm en de rand krijgen nieuwe kleuren in regel 1040. De vertragingslus in 1050 verhoogt de spanning. De regels 1060-1090 PRINTen de relevante meldingen op het scherm. In regel 1090 wordt gevraagd of de speler nog een keer wil spelen. De volgende regels ontvangen en verwerken het antwoord. Bij een testmethode zoals deze accepteert de computer slechts een van de twee verwachte antwoorden. De instructie INKEY$ kijkt of er een toets is ingedrukt. Als het antwoord J (ja) is, begint het spel opnieuw. Is het antwoord N (nee), dan wordt de boodschap TOT ZIENS gePRINT en het spel beëindigd. Geeft u een foutief antwoord, dan wordt dat niet geaccepteerd en ook wanneer u geen antwoord geeft, wordt de vraag opnieuw gesteld.

Het verdient aanbeveling deze methode om fouten te ontdekken te gebruiken bij elke te verwachten INPUT. Zorg ervoor dat de computer niet hangt als een speler een foutief antwoord geeft.

Het leeuwedeel van het spel is af. Nu nog een titelscherm, een paar geluidsroutines en wat kleur erbij en het spel is compleet. We beginnen met het titelscherm.

```
900 REM TITELSCHERM
910 CLS:REM SCHOON SCHERM
920 LOCATE 8,11:PRINT"MAANLANDING"
930 REM GELUID
940 FOR Q=1 TO 255 STEP.5
950 SOUND 0,Q
960 SOUND 7,62
970 SOUND 8,9
980 NEXT Q
990 FOR Q=1 TO 8:SOUND Q,0:NEXT Q
1000 FOR TD=1 TO 1000:NEXT TD
1010 RETURN
```

Dit is een heel simpel titelscherm. Het laat de naam van het spel zien en produceert een korte geluidsexplosie. Als u wilt, kunt u het vervangen door een van de bewegende titelroutines uit het vorige hoofdstuk of door een routine die u zelf hebt gemaakt. Regel 910 wist het scherm. Scherm- en kaderkleuren zijn al ingesteld aan het eind van de initialisatieroutine.

Gekruid geluid

De regels 930-990 zorgen voor het geluid. Regel 940 maakt gebruik van de variabele Q en van een FOR/NEXT-lus, waarin telkens Q met 0,5 wordt verhoogd. Regel 950-970 geven de toon en de geluidssterkte voor kanaal A aan. We laten die toon in regel 950 bepalen door Q. Het resultaat is een steeds hoger wordend geluid. Heeft Q de maximum waarde bereikt, dan springt het programma naar de volgende regel (990) waar het geluid middels een korte routine wordt uitgeschakeld. Deze uitleg is zeer oppervlakkig, maar in een later stadium zullen we het onderwerp 'geluid' uitvoeriger bespreken. Regel 1000 zorgt voor een kleine vertraging zodat de titel niet voorbij zoeft, maar we de tijd hebben om hem te lezen.

```
660 SOUND 0,255
670 SOUND 7,62
680 SOUND 8,9
690 FOR TD=1 TO 10:NEXT TD
700 SOUND 8,0
710 RETURN
```

Dit is de geluidsroutine die in het belangrijkste deel van het spel wordt gebruikt om het motorengebulder van het ruimteschip voort te brengen. Regel 110 roept deze routine op, die de toon voortbrengt en onmiddellijk terugkeert naar de hoofdlus.

```
270 SOUND 3,100
280 SOUND 6,20
290 SOUND 7,47
300 SOUND 9,9
310 FOR TD=1 TO 200:NEXT TD
320 FOR I=0 TO 9:SOUND I,0:NEXT I
330 RETURN
340 SOUND 0,100
```

```
350 SOUND 3,30
360 SOUND 6,15
370 SOUND 7,46
380 SOUND 8,9
390 SOUND 9,9
400 FOR TD=1 TO 100:NEXT TD
410 FOR I=0 TO 9:SOUND I,0:NEXT I
420 RETURN
```

En hier hebben we de overige twee geluidsroutines die in het programma worden gebruikt. De routine in regel 270-330 produceert de klap van de botsing. De tweede routine (regel 340 e.v.) is de routine die een extra ruis aan de klap toevoegt.

```
10 REM MAANLANDING
20 GOSUB 560:REM INITIALISEREN
30 GOSUB 900:REM TITELSCHERM
40 GOSUB 720:REM SCHERM KLAARMAKEN
```

Het enige deel van het programma dat we nog niet hebben besproken, is het begin. Regel 20 roept de initialisatieroutine op. Regel 30 roept de routine op voor het titelscherm en in regel 40 springt het programma naar de routine voor het spelscherm. Vervolgens gaat het programma verder met de hoofdlus.

Hoofdstuk 10

Grafiek uit eigen keuken

We hebben tot nu toe de mogelijkheden van programma's met bewegende grafiek onderzocht. We hebben squash gespeeld en daarna in ons eigen ruimteschip gevlogen. Met de grafische tekens uit de karakterset van de MSX konden we aardig uit de voeten. Maar het wordt tijd dat we onze eisen opschroeven. Zelf grafische afbeeldingen ontwerpen is natuurlijk veel leuker en wat meer is: u kunt elk programma naar eigen inzicht zo aantrekkelijk mogelijk maken. We moesten er maar snel mee beginnen.

We hebben het al eerder gehad over het video-RAM. Dit is het geheugen waarin alles wordt opgeslagen wat te maken heeft met uitvoer op het scherm. Het VRAM speelt een zeer belangrijke rol. Met behulp daarvan kunnen we tekens over het scherm laten rollen en kleuren veranderen. In dit hoofdstuk laten we zien dat we door middel van het VRAM ook onze eigen grafische afbeeldingen kunnen maken. De MSX biedt u de mogelijkheid karakters uit de standaard karakterset te herdefiniëren tot elk gewenst karakter. Laten we eerst eens bekijken hoe de karakterset is opgeslagen. Als u niet bent geïnteresseerd in technische details, lees dan dit hoofdstuk vluchtig door. Ook zonder het naadje van de kous te weten kunt u zelf met grafiek aan de slag.

De karakterset is dus opgeslagen in het VRAM. Elk karakter neemt acht geheugenplaatsen of adressen in beslag. Elke geheugenplaats bestaat uit 8 *bits*. 8 bits die bij elkaar horen noemen we een *byte*. Een bit kunt u zich het beste voorstellen als een schakelaar: hij kan aan en uit zijn. Heeft een bit de waarde 0, dan is hij uit. We zeggen dat de bit is *gewist*. Bij de waarde 1 is hij aan, we noemen dat *gezet*. De waarde van een byte hangt dus af van de toestand van de bits. Onderstaande tekening verduidelijkt dit:

128	64	32	16	8	4	2	1
0	1	0	1	1	0	0	1

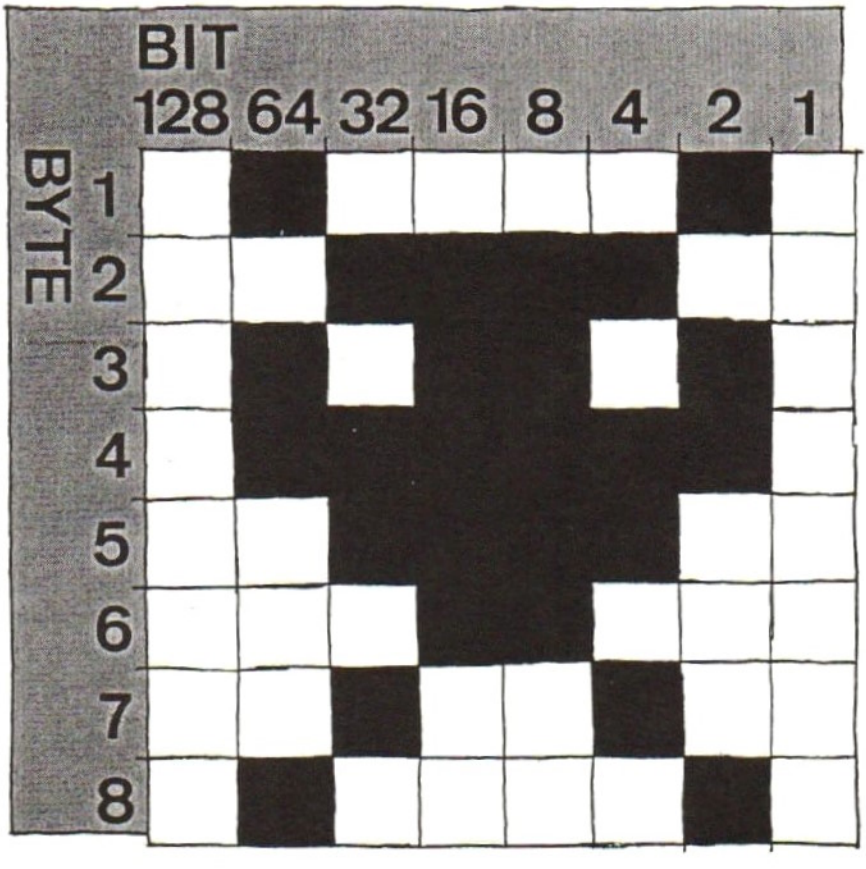
Zoals u ziet zijn alle acht bits genummerd. De computer telt de waarden op van de bits die zijn gezet. In bovenstaande voorbeeld zijn de waarden van de gezette bits 64, 16, 8 en 1. De som daarvan is 89, de waarde van deze byte. Als alle bits gezet zijn, heeft de byte een waarde van 255. Daarom kunt u in een geheugenplaats geen waarde groter dan 255 VPOKEn. Het volgende diagram laat zien hoe acht bytes worden gebruikt om een teken uit de karakterset op te slaan. Het bitpatroon in deze karaktermatrix stelt de letter Y voor.

BYTE \ BIT	128	64	32	16	8	4	2	1	DATA
1	0	1	1	0	0	1	1	0	102
2	0	1	1	0	0	1	1	0	102
3	0	1	1	0	0	1	1	0	102
4	0	0	1	1	1	1	0	0	60
5	0	0	0	1	1	0	0	0	24
6	0	0	0	1	1	0	0	0	24
7	0	0	0	1	1	0	0	0	24
8	0	0	0	0	0	0	0	0	0

Deze matrix laat zien dat het patroon van de bits die zijn gezet de vorm van de letter Y heeft. De acht bytes hebben de volgende waarden: 102, 102, 102, 60, 24, 24, 24, 0.

Nu is het nog maar een kleine stap naar onze eigen grafische karakters. We weten inmiddels hoe de MSX zijn karakters opslaat. Het gemakkelijkst kunt u uw eigen karakters ontwerpen met behulp van de 8*8 rasters achter in het boek. Teken de gewenste figuur en tel per byte de waarden van de gezette bits op, precies als in het voorbeeld van de letter Y.

We geven nog een voorbeeld: een ruimtewezen. Het raster met de gewenste figuur ziet er zo uit:



In dit raster kunt u zien hoe het ruimtewezen is getekend en hoe de bitwaarden zijn opgeteld om de waar-

den voor dit nieuwe teken te verkrijgen. In de eerste byte zijn de bits 2 en 7 gezet: dat is 66.

byte 1 $64 + 2 = 66$
byte 2 $32 + 16 + 8 + 4 = 60$
byte 3 $64 + 16 + 8 + 2 = 90$
byte 4 $64 + 32 + 16 + 8 + 4 + 2 = 126$
byte 5 $32 + 16 + 8 + 4 = 60$
byte 6 $16 + 8 = 24$
byte 7 $32 + 4 = 36$
byte 8 $64 + 2 = 66$

De waarden hebben we gevonden; nu nog een routine om deze waarden op de plaats van de oude waarden te zetten. Daarover gaat de volgende paragraaf, waarin we ook het ruimtemannetje op het scherm toveren.

De hoofdroutine

In de voorgaande paragraaf hebben we gezien hoe een karakter in het VRAM is opgebouwd. Wat we nu gaan doen is die karakters veranderen door de bytes andere waarden te geven. Allereerst is het heel belangrijk te weten waar de verschillende karakters zijn opgeslagen. Dat is afhankelijk van het SCREEN waarin u zich bevindt. In SCREEN 0 is het beginadres van de karakterset 2048; in SCREEN 1, 2 en 3 is het beginadres 0. In SCREEN 0 zijn bijvoorbeeld de waarden van karakter 0 opgeslagen in de adressen 2048-2055. In SCREEN 1, 2 en 3 zijn dat de adressen 0-7.

Vervolgens hebben we een routine nodig waarmee u elk willekeurig karakter kunt herdefiniëren tot het door u gewenste karakter. In deze routine maken we gebruik van een READ-instructie waarmee we de gewenste waarden uit DATA-regels kunnen inlezen. De instructie VPOKE hebben we weer nodig om de waarden in het geheugen van de computer te zetten.

```
10 REM CUSTOM CHARACTERS
20 SCREEN 1
30 CLS
40 FOR N=65 TO 66
50 FOR M=0 TO 7
60 READ A
70 VPOKE 0+N*8+M,A
```

```
80 NEXT M
90 NEXT N
100 LOCATE 11,12
110 PRINT "A  B"
120 DATA 0,126,126,126,126,126,126,0
130 DATA 255,129,129,129,129,129,129,255
```

De grote lus

We maken gebruik van het basisadres (in dit geval 0 omdat we ons in SCREEN 1 bevinden) en van twee FOR/NEXT-lussen. De variabele N geeft het karakter aan dat we gaan herdefiniëren (in dit geval de karakters 65 en 66). De variabele M is de teller voor de acht verschillende bytes van het karakter. De gegevens voor de nieuwe karakters staan in de DATA-regels 120 en 130. De regels 100 en 110 dienen om de nieuwe karakters op het scherm te PRINTen.

In feite hebben we met dit programma de MSX voor de gek gehouden. We gaven opdracht om een A en een B op het scherm te PRINTen, maar omdat we de letters veranderd hebben tot een blokje en een kadertje, werden niet de A en de B maar de nieuwe karakters gePRINT. Dat is precies wat we nodig hebben. Hoewel we een karakter hebben veranderd, wordt het door de computer nog steeds als het oorspronkelijke karakter behandeld. Het kan op het scherm worden gePRINT, het kan in het schermgeheugen worden geVPOKEt, en als we een VPOKE op het betreffende schermadres loslaten, krijgen we de waarde van het oorspronkelijke karakter (in ons voorbeeld de waarde 65 of 66). Dit opent nieuwe perspectieven voor eigen grafische programma's. U kunt alle eigenschappen gebruiken die we in vorige hoofdstukken hebben besproken, maar u bent niet gebonden aan de standaard karakterset. De ingrediënten voor grafiek uit eigen keuken zijn nu allemaal aanwezig.

In het volgende programmaatje laten we zien dat karakters niet alleen in SCREEN 1 maar ook in SCREEN 0 opnieuw kunnen worden gedefinieerd. Het verschil met het vorige programma is dat het nieuwe karakter maar voor driekwart gedeelte (of om precies te zijn zes achtste deel) te zien is. Dat komt doordat in SCREEN 0 per karakter maar 6 punten beschikbaar zijn. We zetten het ruimtemannetje van de tekening enkele pagina's terug nog eens op het scherm.

```
10 REM GRAFIEK
20 SCREEN 0
30 CLS
40 FOR N=65 TO 65
50 FOR M=0 TO 7
60 READ A
70 VPOKE 2048+N*8+M,A
80 NEXT M
90 NEXT N
100 LOCATE  15,12
110 PRINT "A"
120 DATA 66,60,90,126,60,24,36,66
```

We hebben een programma geschreven dat 33 karakters herdefinieert in figuren die u kunt gebruiken in eigen programma's. Typ elke regel nauwkeurig in, vooral de DATA-regels. RUN het programma en bekijk de nieuwe karakters maar eens goed.

```
10 REM CUSTOM CHARACTERS
20 KEY OFF
30 SCREEN 1
40 CLS
50 FOR N=65 TO 98
60 FOR M=0 TO 7
70 READ A
80 VPOKE N*8+M,A
90 NEXT M
100 NEXT N
110 LOCATE 0,4:PRINT"A  B  C  D  E  F  G  H
I  J"
120 LOCATE 0,6:PRINT"K  L  M  N  O  P  Q  R
S  T"
130 LOCATE 0,8:PRINT"U  V  W  X  Y  Z  [  \
]  ^"
140 LOCATE 0,10:PRINT"_  `  a  b"
150 REM INDRINGER 1
160 DATA 66,60,90,126,60,24,36,66
170 REM INDRINGER 2
180 DATA 60,126,219,255,102,60,90,129
190 REM INDRINGER 3
200 DATA 66,36,189,165,255,60,36,231
210 REM INDRINGER 4
220 DATA 62,42,62,28,8,119,65,65
```

```
230 REM PIJLEN
240 REM NAAR LINKS
250 DATA 16,48,112,255,255,112,48,16
260 REM NAAR RECHTS
270 DATA 8,12,14,255,255,14,12,8
280 REM OMHOOG
290 DATA 24,60,126,255,24,24,24,24
300 REM PACMAN-TEKENS
310 REM PACMAN RECHTS
320 DATA 60,127,252,240,240,252,127,60
330 REM PACMAN LINKS
340 DATA 60,254,63,15,15,63,254,60
350 REM PACMAN OMHOOG
360 DATA 66,66,231,231,255,255,126,60
370 REM PACMAN OMLAAG
380 DATA 60,126,255,255,231,231,66,66
390 REM GEEST
400 DATA 56,124,214,214,254,254,170,170
410 REM RACE-AUTO
420 REM AUTO NAAR RECHTS
430 DATA 0,238,68,255,255,68,238,0
440 REM AUTO OMLAAG
450 DATA 90,126,90,24,90,126,90,24
460 REM AUTO NAAR LINKS
470 DATA 0,119,34,255,255,34,119,0
480 REM AUTO OMHOOG
490 DATA 24,90,126,90,24,90,126,90
500 REM VLIEGTUIG
510 REM VLIEGTUIG OMHOOG
520 DATA 24,24,60,126,255,24,24,60
530 REM VLIEGTUIG NAAR RECHTS
540 DATA 16,24,156,255,255,156,24,16
550 REM VLIEGTUIG OMLAAG
560 DATA 60,24,24,255,126,60,24,24
570 REM VLIEGTUIG NAAR LINKS
580 DATA 8,24,57,255,255,57,24,8
590 REM ZEILBOOT
600 DATA 8,24,60,126,8,255,126,60
610 REM MAN 1
620 DATA 56,56,16,124,16,56,68,68
630 REM BOOM
640 DATA 24,24,60,24,126,60,255,24
650 REM EXPLOSIE
660 DATA 145,82,0,192,3,0,74,137
```

```
670 REM RUIMTESCHIP IN ALLE ACHT POSITIES GEZ
IEN
680 DATA 24,24,36,36,66,90,165,195
690 DATA 3,13,50,194,52,20,8,8
700 DATA 192,176,76,35,35,76,176,192
710 DATA 8,8,20,52,194,50,13,3
720 DATA 195,165,90,66,36,36,24,24
730 DATA 16,16,40,44,67,76,176,192
740 DATA 3,13,50,196,196,50,13,3
750 DATA 192,176,76,67,44,40,16,16
760 REM LEEG TEKEN
770 DATA 0,0,0,0,0,0,0,0
780 REM MAN 2
790 DATA 24,16,60,90,153,36,66,99
```

Het eerste deel van het programma kende u al. Het is de routine die we ook hebben gebruikt om de letter A te herdefiniëren tot een ruimtewezen. Het basisadres dat we bij het vorige programma gebruikten, laten we nu weg. We werken immers in SCREEN 1 en het basisadres in SCREEN 1 is 0. Als u het programma RUNt en de foutmelding 'Out of DATA' verschijnt, hebt u het getal in de FOR/NEXT-lus niet correct overgenomen of u hebt enkele DATA weggelaten.

In dit programma nemen de DATA-regels veel geheugenruimte in beslag. Een aantal REM-regels is toegevoegd om elk karakter te verklaren en voor elk karakter is een aparte DATA-regel gebruikt. Op deze manier kunnen wel gemakkelijk fouten worden opgezocht. Als een van de DATA niet correct was, is dit heel eenvoudig na te gaan door de karakters van boven naar beneden te onderzoeken. Als u genoeg geheugenruimte hebt, kan het erg handig zijn DATA-regels op deze manier te schrijven.

U heeft wellicht opgemerkt dat enkele karakters meerdere keren voorkomen. Het enige verschil is dat ze in verschillende richtingen kijken. Dit is een fraaie mogelijkheid bij eigen karakters. Als u bijvoorbeeld een spel wilt schrijven waarin een joystick wordt gebruikt om een ruimteschip over het scherm te bewegen kunt u de acht ruimteschipvarianten uit het laatste programma gebruiken. Het schip vliegt dan altijd met de neus in de gewenste richting.

Hoofdstuk 11

Reverse tekens

In dit hoofdstuk borduren we nog even voort op de mogelijkheid om karakters opnieuw te definiëren. De MSX kent standaard niet de mogelijkheid om karakters reverse weer te geven. Reverse wil zeggen dat karakters op het scherm worden gezet alsof ze 'omgekeerd' zijn weergegeven: wat wit is wordt zwart en wat zwart is wit. Fotografen onder ons klinkt de term 'diapositief' waarschijnlijk bekender in de oren. Nu bestaat bij de MSX de mogelijkheid om een gedeelte van de karakterset uit te lezen, deze 'om te keren' en op de plaats van andere karakters weer weg te schrijven.

Voordat we uit de doeken doen hoe dit precies in zijn werk gaat, geven we eerst een programma waarin gebruik wordt gemaakt van reverse karakters.

```
10 REM REVERSE
20 SCREEN 0
30 KEY OFF
40 FOR N=65 TO 90
50 FOR M=0 TO 7
60 B=VPEEK(2048+N*8+M)
70 C=NOT(B)
80 C$=RIGHT$(BIN$(C),8)
90 C=VAL("&B"+C$)
100 VPOKE 2048+256+N*8+M,C
110 NEXT M
120 NEXT N
130 PRINT"U HEBT NU DE KEUS TUSSEN GEWONE":PR
INT"TEKENS OF reverse tekens"
140 PRINT"DE kleine letters ZIJN IN reverse":
PRINT"hoofdletters VERANDERD"
```

In het programma komen de eerste regels u zeker bekend voor: de REM-regel, de schermkeuze en het uitschakelen van de balk met functietoetsen. Bij regel 60 ziet alles er iets anders uit: er wordt niet geVPOKEt maar geVPEEKt. Hier lezen we de adressen van de hoofdletters uit, in plaats van dat we er iets anders neerzetten. We kennen de waarde van het adres toe aan B en maken dan gebruik van de logische operator NOT om elke bit die niet gezet is te zetten en elke bit die wel gezet is te resetten. Een voorbeeld:

NOT(&B10001101)

wordt

&B1111111101110010

Uit dit voorbeeld wordt duidelijk waarom we in regel 90 van het binaire getal de rechter 8 bits gebruiken. NOT kan niet alleen 8-bits getallen aan maar ook 16-bits getallen en de uitvoer van NOT is altijd een 16-bits getal. De denkbeeldige 8 nullen die voor het 8-bits getal staan, worden door NOT omgezet in enen.

Vervolgens zetten we in regel 100 het zo verkregen binaire getal weer om in een decimaal getal en VPOKEn dat op de plaats van dezelfde, maar dan kleine letter. Het resultaat is dat wanneer u een kleine letter indrukt, u wel de juiste letter op het scherm krijgt, maar dan als reverse hoofdletter. De regels 130 en 140 zijn bedoeld om het effect van het programma te laten zien.

Hoofdstuk 12

Een adempauze

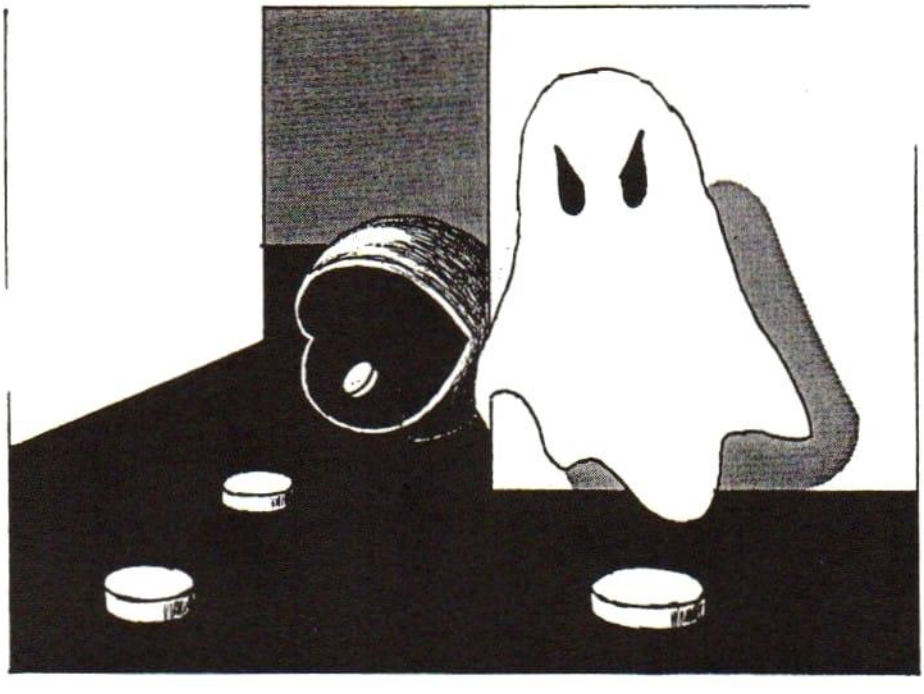
Voordat we overgaan tot een nieuw onderwerp, zullen we zien hoeveel lol we kunnen beleven aan eigen grafische ontwerpen. In dit hoofdstuk introduceren we een prachtig spel: HAPPY HAPPER. Het professionele karakter van dit spel is grotendeels te danken aan het ingenieuze gebruik van grafiek. In dit programma komt u routines tegen die we al hebben besproken, maar ook hele nieuwe programmeertrucs en ideeën. Speel het maar eens, u beleeft er vast veel plezier aan. We zullen na afloop het programma eens doorlichten om te zien hoe het tot stand is gekomen.

De bedoeling van het spel is de Happer door een van de twee doolhoven te loodsen. De Happer moet pilletjes opeten en voorkomen dat hij zelf wordt opgegeten door de monsters. Als hij 30 pilletjes heeft verorberd, gaat hij verder op het volgende scherm. Stuit hij op een monster, dan is de Happer de klos. Als het een supermonster betreft, kan hij zelfs twee keer worden verslonden. Maar de monsters kunnen ook ten prooi vallen aan de Happer, wanneer deze tenminste eerst zijn wonderpil heeft geslikt. Een toon maakt u erop attent dat de Happer de pil inneemt. In de volgende seconden kan hij zelf monsters vangen en verslinden. U krijgt drie Happers. Als ze alle drie door de monsters zijn verzwolgen, is het spel afgelopen.

Er is nog een mogelijkheid waarop we nog niet hebben gewezen. Als u in de val zit en niet kunt ontsnappen, kunt u de drukknop gebruiken. U kunt dan door een muur van de doolhof lopen. Daarvoor moet de Happer met zijn snuit in de gewenste richting tegen de muur gaan staan. U kunt deze mogelijkheid hoogstens twee keer per scherm gebruiken.

Als het spel wordt geRUNd, verzoekt het programma u te wachten totdat het spel is geladen. Na een korte pauze wordt de titelpagina gePRINT. We hebben een slimme routine gebruikt om wat beweging in deze pagina te krijgen. Als de titelpagina is gewist, kunt u kiezen uit doolhof 1 en 2. Als u uw keuze kenbaar heeft gemaakt, verschijnt de doolhof op het scherm. Daaronder staat de benodigde informatie: hoeveel Happers u nog over heeft en hoeveel pilletjes de Happer al heeft opgegeten. U kunt de score aflezen en rechtsonder staat het belangrijkste: de status van de Happer. Twee spookjes tussen naar buiten gekeerde haakjes betekent dat een ontmoeting met een monster voor uw Happer fataal is. Als de Happer zijn wonderpil slikt, worden de haakjes omgedraaid. De Happer kan nu veilig op monsterjacht gaan. Hou tijdens het spel de Happer-status goed in de gaten. Twee spookjes tussen twee naar buiten gekeerde haakjes betekent oppassen en wegwezen.

Happy Happer is een lang programma met twee grote PRINT-onderdelen, waarin de doolhoven worden gePRINT. Neem elke regel exact over. Het is van groot belang dat u bij de PRINT-opdrachten het juiste aantal karakters (CODE SHIFT H) en spaties typt. Doet u dat niet dan verschijnt de doolhof niet goed op het scherm.



```
10 REM HAPPY HAPPER
20 GOSUB 1720
30 GOSUB 2380
40 GOSUB 1230
50 P1=10:P2=50
60 GOSUB 450
70 PM=1:X=5:Y=9:W=21:E=10:D=1:DI=2:E2=8:W2=9:
MN=3:CN=102:NU=0:LR=MN
80 GOSUB 1060
90 GOSUB 1130
100 JV=STICK(1)
110 FR=STRIG(1)
120 IF VU=100 THEN IO=IO+1:IF IO=20 THEN VU=0
:GOSUB 1060
130 IF JV=1 THEN Y=Y-1:CN=99
140 IF JV=5 THEN Y=Y+1:CN=100
150 IF JV=3 THEN X=X+1:CN=102
160 IF JV=7 THEN X=X-1:CN=101
170 IF FR<>0 THEN AN=AN+1:IF AN<3 THEN PN=3:G
OTO 270
180 IF NU>29 THEN GOTO 2540
190 CRE=INT(RND(1)*40+1):IF CRE<>1 THEN 220
200 QW=INT(RND(1)*28+2):QR=INT(RND(1)*17+1):I
F VPEEK(6144+QW+QR*32)=176 THEN GOTO 200
210 VPOKE 6144+QW+QR*32,160
220 CR=VPEEK(6144+X+Y*32):IF CR=176 THEN X=R:
Y=K
230 IF CR=168 THEN SC=SC+40:VU=100:IO=0:GOSUB
 1050:GOSUB 1940
240 IF CR=160 THEN SC=SC+10:NU=NU+1:GOSUB 201
0
250 IF CR=104 THEN GOSUB 1070
260 IF CR=184 THEN GOSUB 1150
270 GOSUB 880
280 GOSUB 960
290 RE=INT(RND(1)*10+1):IF RE>8 THEN D=INT(RN
D(1)*4+1)
300 PM=PM*-1:IF PM=1 THEN PN=CN:GOTO 320
310 PN=103
320 IF R=W AND K=E THEN GOSUB 1070
330 IF R=W2 AND K=E2 THEN GOSUB 1150
340 VPOKE 6144+Q+O*32,32:VPOKE 6144+Q2+O2*32,
32:VPOKE 6144+R+K*32,32
350 VPOKE 6144+X+Y*32,PN:VPOKE 6144+W+E*32,10
```

```
4:VPOKE 6144+W2+E2*32,184
360 LOCATE 9,21:PRINT SC
370 LOCATE 23,20:PRINT NU
380 CRE=INT(RND(1)*40+1):IF CRE<>1 THEN 410
390 QW=INT(RND(1)*28+2):QR=INT(RND(1)*17+1):I
F VPEEK(6144+QW+QR*32)=176 THEN GOTO 390
400 VPOKE 6144+QW+QR*32,168
410 R=X:K=Y:Q=W:O=E:Q2=W2:O2=E2
420 IF W<1 THEN W=1
430 IF W2<1 THEN W2=1
440 GOTO 100
450 REM TEKEN HET DOOLHOF
460 ON LV GOTO 470,1410
470 COLOR 15,1,1
480 CLS
490 VPOKE KL+12,129
500 VPOKE KL+13,65
510 VPOKE KL+20,33
520 VPOKE KL+22,235
530 VPOKE KL+23,161
540 VPOKE KL+24,209
550 CLS
555 REM ALLE PUNTEN MOETEN VERVANGEN WORDEN D
OOR CODE y
560 PRINT".............................."
570 PRINT".    .   ..       ..   .    ."
580 PRINT".   . . .. .   . .. . .    ."
590 PRINT".   . . .  .   .  . . .    ."
600 PRINT". ... .    .   .    . .... ."
610 PRINT".     ......   ......       ."
620 PRINT".  ..                  ...  ."
630 PRINT".    .    ... ...    .     ."
640 PRINT". .....  ...   ...  ...... ."
650 PRINT".       ...     ...        ."
660 PRINT". .....  ...   ...  ...... ."
670 PRINT".    .    ... ...    .     ."
680 PRINT".  ..                  ...  ."
690 PRINT".     ......   ......       ."
700 PRINT". ... .    .   .    . .... ."
710 PRINT".   . . .  .   .  . . .    ."
720 PRINT".   . . .. .   . .. . .    ."
730 PRINT".   .   ..       ..   .    ."
740 PRINT".............................."
750 PRINT
```

```
760 PRINT"HAPPERS =       AANTAL ="
770 PRINT"  SCORE =       STATUS ="
780 REM TEKEN PILLETJES
790 FOR N=1 TO P1
800 X=INT(RND(1)*28+2):Y=INT(RND(1)*17+1):F=V
PEEK(6144+X+Y*32):IF F=176 THEN GOTO 800
810 VPOKE 6144+X+Y*32,168
820 NEXT N
830 FOR P=1 TO P2
840 X=INT(RND(1)*28+2):Y=INT(RND(1)*17+1):F=V
PEEK(6144+X+Y*32):IF F=176 THEN GOTO 840
850 VPOKE 6144+X+Y*32,160
860 NEXT P
870 RETURN
880 REM EERSTE MONSTER BEWEGEN
890 IF D=1 THEN E=E-1
900 IF D=2 THEN E=E+1
910 IF DI=3 THEN W=W-1
920 IF DI=4 THEN W=W+1
930 I=VPEEK(6144+W+E*32):IF I=176 THEN 950
940 RETURN
950 D=INT(RND(1)*4+1):W=Q:E=O:GOTO 940
960 REM TWEEDE MONSTER BEWEGEN
970 IF DI=1 THEN E2=E2-1
980 IF DI=2 THEN E2=E2+1
990 IF DI=3 THEN W2=W2-1
1000 IF DI=4 THEN W2=W2+1
1010 I=VPEEK(6144+W2+E2*32):IF I=176 THEN 103
0
1020 RETURN
1030 DI=INT(RND(1)*4+1):W2=Q2:E2=O2:GOTO 1020
1040 REM STATUS AANGEVEN
1050 LOCATE 23,21:PRINT"(";CHR$(104);CHR$(184
)")":RETURN
1060 LOCATE 23,21:PRINT")";CHR$(104);CHR$(184
)"(":RETURN
1070 REM BOTS-ROUTINE (EERSTE MONSTER)
1080 IF VU=100 THEN E=9:W=21:SC=SC+200:D=1:GO
SUB 2080:RETURN
1090 GOSUB 2200
1100 VPOKE 6144+R+K*32,192:FOR TD=1 TO 200:NE
XT TD:AN=0
1110 X=5:Y=9:CN=102:MN=MN-1:LR=MN:IF MN<1 THE
N GOSUB 2290:GOTO 1300
```

```
1120  LOCATE 9,20:PRINT"    "
1130 LOCATE 9,20:FOR T=1 TO MN-1:PRINT CHR$(1
02);:NEXT T
1140 RETURN
1150 REM BOTS-ROUTINE (TWEEDE MONSTER)
1160 IF VU=100 THEN E2=9:W2=21:SC=SC+200:DI=2
:GOSUB 2080:RETURN
1170 GOSUB 2200
1180 VPOKE 6144+R+K*32,192:FOR TD=1 TO 200:NE
XT TD:AN=0
1190 X=5:Y=9:CN=102:MN=MN-1:LR=MN:IF MN<1 THE
N GOSUB 2290:GOTO 1300
1200 GOTO 1000
1210 RETURN
1220 END
1230 REM DOOLHOF BEPALEN
1240 CLS
1250 LOCATE 5,12:PRINT"WELK DOOLHOF 1/2"
1260 A$=INKEY$
1270 IF A$<>"1" AND A$<>"2" THEN 1260
1280 LV=VAL(A$)
1290 RETURN
1300 REM EINDE SPEL
1310 CLS
1320 COLOR 1,9,9
1330 LOCATE 4,10:PRINT CHR$(102);"JE BENT ER
GEWEEST";CHR$(101)
1340 LOCATE 4,12:PRINT"JE SCOORDE: ";SC
1350 FOR TD=1 TO 2000:NEXT TD
1360 LOCATE 4,14:PRINT"NOG EEN SPEL (J/N)"
1370 A$=INKEY$
1380 IF A$="J" THEN RUN
1390 IF A$<>"N" THEN GOTO 1370
1400 END
1410 CLS
1420 COLOR 1,15,15
1430 VPOKE KL+12,79
1440 VPOKE KL+13,143
1450 VPOKE KL+20,207
1460 VPOKE KL+22,27
1470 VPOKE KL+23,207
1480 VPOKE KL+24,31
1485 REM ALLE PUNTEN MOETEN VERVANGEN WORDEN
DOOR CODE y
```

```
1490 PRINT"............................."
1500 PRINT".    .       .        .    ."
1510 PRINT". .. .                . .. ."
1520 PRINT". .. .  ............  . .. ."
1530 PRINT".                          ."
1540 PRINT"....  ..     .      ..  ...."
1550 PRINT".     ...          ...     ."
1560 PRINT". .         .. ..         . ."
1570 PRINT". . .     ..   ..      . . ."
1580 PRINT".   .    ..     ..     .   ."
1590 PRINT". . .     ..   ..      . . ."
1600 PRINT". .         .. ..         . ."
1610 PRINT".     ...          ...     ."
1620 PRINT"....  ..     .      ..  ...."
1630 PRINT".                          ."
1640 PRINT". .. .  ............  . .. ."
1650 PRINT". .. .                . .. ."
1660 PRINT".    .       .        .    ."
1670 PRINT"............................."
1680 PRINT
1690 PRINT"HAPPERS =      AANTAL ="
1700 PRINT"  SCORE =      STATUS ="
1710 GOTO 780
1720 REM INITIALISATIE
1730 L=RND(-TIME)
1740 SCREEN 1
1750 KEY OFF
1760 CLS
1770 COLOR 1,3,3
1780 KL=8192
1790 LOCATE 8,11:PRINT"EVEN GEDULD"
1800 LOCATE 8,12:PRINT STRING$(11,CHR$(188))
1810 FOR N=99 TO 104
1820 FOR M=0 TO 7
1830 READ A
1840 VPOKE N*8+M,A
1850 NEXT M
1860 NEXT N
1870 FOR N=160 TO 192 STEP 8
1880 FOR M=0 TO 7
1890 READ A
1900 VPOKE N*8+M,A
1910 NEXT M
1920 NEXT N
```

```
1930 RETURN
1940 REM KRACHTPILLEN
1950 SOUND 0,100
1960 SOUND 7,62
1970 SOUND 8,14
1980 FOR TD=1 TO 25:NEXT TD
1990 SOUND 7,63
2000 RETURN
2010 REM KLEINE PILLEN
2020 SOUND 0,50
2030 SOUND 7,62
2040 SOUND 8,14
2050 FOR TD=1 TO 10:NEXT TD
2060 SOUND 7,63
2070 RETURN
2080 REM MONSTER DOOD
2090 SOUND 0,25
2100 SOUND 7,62
2110 SOUND 8,16
2120 SOUND 11,1
2130 SOUND 12,1
2140 SOUND 13,12
2150 FOR TD=1 TO 250:NEXT TD
2160 SOUND 7,63
2170 SOUND 1,0
2180 SOUND 8,15
2190 RETURN
2200 REM HAPPER DOOD
2210 SOUND 0,255
2220 SOUND 1,255
2230 SOUND 7,62
2240 SOUND 8,15
2250 FOR TD=1 TO 90:NEXT TD
2260 SOUND 7,63
2270 SOUND 1,0
2280 RETURN
2290 REM EIND-MUZIEK
2300 FOR Z=1 TO 255
2310 SOUND 0,Z
2320 SOUND 7,62
2330 SOUND 8,15
2340 FOR TD=1 TO 5:NEXT TD
2350 NEXT Z
2360 SOUND 7,63
```

```
2370 RETURN
2380 REM TITELPAGINA
2390 CLS
2400 COLOR 13,1,1
2410 VPOKE KL+12,97
2420 LOCATE 8,11
2430 PRINT"HAPPY HAPPER"
2440 FOR TD=1 TO 500:NEXT TD
2450 FOR Z=0 TO 31
2460 VPOKE 6144+352+Z,102
2470 FOR TD=1 TO 50:NEXT TD
2480 VPOKE 6144+352+Z,103
2490 FOR TD=1 TO 20:NEXT TD
2500 VPOKE 6144+352+Z,32
2510 NEXT Z
2520 COLOR 1,8,8
2530 RETURN
2540 REM BEGIN MET EEN NIEUW DOOLHOF
2550 IF LV=1 THEN LV=2 ELSE LV=1
2560 LOCATE 0,22:PRINT"PRIMA, HET VOLGENDE DO
OLHOF"
2570 FOR TD=1 TO 2000:NEXT TD
2580 GOSUB 450
2590 PM=1:X=5:Y=9:W=21:E=10:D=1:DI=2:E2=8:W2=
9:CN=102:NU=0:LR=MN:AN=0
2600 P1=P1-2:P2=P2-2
2610 GOTO 80
2620 REM DATA HAPPERS
2630 DATA 66,102,231,231,255,126,60,0
2640 DATA 0,60,126,255,231,231,102,66
2650 DATA 56,252,124,14,14,124,252,56
2660 DATA 28,63,126,112,112,126,63,28
2670 DATA 28,62,127,127,127,127,62,28
2680 REM DATA SPOOK
2690 DATA 24,60,126,90,90,126,126,90
2700 REM DATA PILLETJES
2710 DATA 0,0,0,24,24,0,0,0
2720 REM DATA MUUR
2730 DATA 0,0,24,60,60,24,0,0
2740 REM DATA MUUR
2750 DATA 255,195,165,153,153,165,195,255
2760 REM DATA SPOOK
2770 DATA 24,60,126,90,90,126,126,90
2780 REM DATA EXPLOSIE
2790 DATA 34,128,42,32,156,66,36,146
```

Het programma begint met een initialisatieroutine:

```
1720 REM INITIALISATIE
1730 L=RND(-TIME)
1740 SCREEN 1
1750 KEY OFF
1760 CLS
1770 COLOR 1,3,3
1780 KL=8192
1790 LOCATE 8,11:PRINT"EVEN GEDULD"
1800 LOCATE 8,12:PRINT STRING$(11,CHR$(188))
1810 FOR N=99 TO 104
1820 FOR M=0 TO 7
1830 READ A
1840 VPOKE N*8+M,A
1850 NEXT M
1860 NEXT N
1870 FOR N=160 TO 192 STEP 8
1880 FOR M=0 TO 7
1890 READ A
1900 VPOKE N*8+M,A
1910 NEXT M
1920 NEXT N
1930 RETURN
```

De routine PRINT eerst een melding, zodat u zich niet afvraagt wat er allemaal gaande is nadat u RUN heeft ingetypt en op RETURN gedrukt. In de regels 1810-1860 en 1870-1920 worden de karakters opnieuw gedefinieerd. In feite is er twee keer dezelfde routine gebruikt, om straks de verschillende karakters een eigen kleur te kunnen geven.

Doordat we bij het herdefiniëren niet bij de letter A begonnen zijn, kunnen naast de nieuwe karakters ook de gewone letters en cijfers worden gebruikt. Dat betekent meer armslag bij het ontwerpen van schermpagina's. Onthoud dit als stelregel: heeft u weinig nieuwe karakters nodig, laat dan alfabet en cijfers intact en gebruik andere tekens om uw eigen fantasie op bot te vieren.

Nadat het programma is geïnitialiseerd, wordt het titelscherm opgeroepen. De titelpagina maakt gebruik van een van de nieuwe tekens, namelijk een Happer. De titel verschijnt op het scherm en wordt even later door de Happer verzwolgen. In de titelroutine maken we weer gebruik van de bekende formule om een karakter over het scherm te

bewegen. De volgende routine vraagt in welke doolhof u zich wilt begeven:

```
1230 REM DOOLHOF BEPALEN
1240 CLS
1250 LOCATE 5,12:PRINT"WELK DOOLHOF 1/2"
1260 A$=INKEY$
1270 IF A$<>"1" AND A$<>"2" THEN 1260
1280 LV=VAL(A$)
1290 RETURN
```

Deze routine gebruikt de VAL-functie om te testen of de invoer geldig is. VAL (value) stelt vast welke waarde een string heeft. Wanneer de VAL-waarde van LV$ niet gelijk is aan 1 of 2, gaat het programma terug naar regel 1260 en vraagt om nieuwe invoer. Bij een geldige invoer wordt de waarde van LV$ aan variabele LV toegekend. In regel 460 wordt de waarde van LV getest en de overeenkomstige doolhofroutine aangeroepen. Als de doolhof op het scherm is gePRINT, de pilletjes en wonderpillen op hun plaats staan en de relevante informatie onder de doolhof is gePRINT, keren we terug naar regel 70 waar aan verschillende variabelen waarden worden toegekend. Via GOSUB 1060 en GOSUB 1130 in regel 80 en 90 worden de status en het resterende aantal Happers op de juiste plaats onder de doolhof gePRINT.

De hoofdlus

We zijn nu bij de hoofdlus van het spel aangeland. Deze lus begint met de routine die de invoer accepteert van de joystick. STICK(1) leest de richting van de joystick uit en STRIG(1) controleert of de vuurknop al of niet is ingedrukt. Heeft u geen joystick, verander dan de beide enen in nullen. Dan kunt u de Happer met de cursorbesturingstoetsen besturen in plaats van met de joystick. De spatiebalk gebruikt u dan als vuurknop.

```
100 JV=STICK(1)
110 FR=STRIG(1)
120 IF VU=100 THEN IO=IO+1:IF IO=20 THEN VU=0
:GOSUB 1060
130 IF JV=1 THEN Y=Y-1:CN=99
140 IF JV=5 THEN Y=Y+1:CN=100
150 IF JV=3 THEN X=X+1:CN=102
160 IF JV=7 THEN X=X-1:CN=101
170 IF FR<>0 THEN AN=AN+1:IF AN<3 THEN PN=3:G
OTO 270
```

Met regel 100 en 110 kunnen de joystick-waarden worden gelezen. Regel 120 hoort eigenlijk niet bij deze routine. Hij geeft de tijd aan gedurende welke onze Happer de monsters kan opeten. Wilt u het spel moeilijker maken, kort deze tijd dan in. De regels 130-160 testen en verwerken de invoer van de joystick. De variabele CN wordt gebruikt om de Happy Happer af te beelden. De waarden van CN staan voor vier verschillende tekens die overeenkomen met de richtingen waarin de Happer kan kijken. Regel 170 test of de knop van de joystick is ingedrukt. Is dat het geval dan wordt AN met 1 verhoogd. Variabele AN houdt bij hoe vaak u de muurroutine hebt gebruikt. Als de knop wordt ingedrukt en AN is kleiner dan 3, dan wordt de routine 'Loop door de muur' aangeroepen.

Nu volgt een korte beschrijving van de rest van de hoofdlus.

```
180 IF NU>29 THEN GOTO 2540
190 CRE=INT(RND(1)*40+1):IF CRE<>1 THEN 220
200 QW=INT(RND(1)*28+2):QR=INT(RND(1)*17+1):I
F VPEEK(6144+QW+QR*32)=176 THEN GOTO 200
210 VPOKE 6144+QW+QR*32,160
220 CR=VPEEK(6144+X+Y*32):IF CR=176 THEN X=R:
Y=K
230 IF CR=168 THEN SC=SC+40:VU=100:IO=0:GOSUB
 1050:GOSUB 1940
240 IF CR=160 THEN SC=SC+10:NU=NU+1:GOSUB 201
0
250 IF CR=104 THEN GOSUB 1070
260 IF CR=184 THEN GOSUB 1150
270 GOSUB 880
280 GOSUB 960
290 RE=INT(RND(1)*10+1):IF RE>8 THEN D=INT(RN
D(1)*4+1)
300 PM=PM*-1:IF PM=1 THEN PN=CN:GOTO 320
310 PN=103
320 IF R=W AND K=E THEN GOSUB 1070
330 IF R=W2 AND K=E2 THEN GOSUB 1150
340 VPOKE 6144+Q+O*32,32:VPOKE 6144+Q2+O2*32,
32:VPOKE 6144+R+K*32,32
350 VPOKE 6144+X+Y*32,PN:VPOKE 6144+W+E*32,10
4:VPOKE 6144+W2+E2*32,184
360 LOCATE 9,21:PRINT SC
370 LOCATE 23,20:PRINT NU
380 CRE=INT(RND(1)*40+1):IF CRE<>1 THEN 410
390 QW=INT(RND(1)*28+2):QR=INT(RND(1)*17+1):I
```

```
F VPEEK(6144+QW+QR*32)=176 THEN GOTO 390
400 VPOKE 6144+QW+QR*32,168
410 R=X:K=Y:Q=W:O=E:Q2=W2:O2=E2
420 IF W<1 THEN W=1
430 IF W2<1 THEN W2=1
440 GOTO 100
```

Regel 180 test of de Happy Happer al 30 pilletjes heeft opgegeten. Als dat het geval is, begint het spel met de volgende doolhof. Regel 190 laat met een kans van 1 op 40 een nieuw pilletje verschijnen. In regel 200 wordt gekeken waar het pilletje kan komen te staan en in regel 210 wordt de nieuwe pil op het scherm geVPOKEt. De regels 220-240 testen wat zich op het pad van de Happer bevindt en roepen de overeenkomstige routines aan. Regel 250 test of de Happer een wonderpil heeft opgegeten. Is dit het geval dan wordt de score verhoogd en worden de monsters kwetsbaar doordat de betreffende variabelen de kwetsbaarheidswaarde 100 krijgen. Als de Happer een gewone pil slikt, telt regel 240 die op bij het aantal geslikte pillen en verhoogt tevens de score. De regels 270-280 roepen de routines op die de monsters bewegen. Elk monster krijgt daarna een kans van 1 op 5 om van richting te veranderen (regel 290). De regels 300-310 bepalen of de Happer op het scherm moet verschijnen met zijn bek open of gesloten.

Het programma controleert vervolgens (regel 320-330) of een van de monsters de Happer heeft gevangen. Regel 340 gebruikt een bekende formule om na alle figuren spaties te VPOKEn. Vervolgens keert dezelfde formule in regel 350 terug om de Happy Happer en de twee monsters op het scherm te VPOKEn. In de regels 360-370 worden de score en het aantal geslikte pillen gePRINT. De regels 380-390 bepalen de kans dat er een nieuwe wonderpil verschijnt. Regel 410 houdt alle variabelen bij die worden gebruikt om de figuren over het scherm te bewegen. Dit omdat de regel die een spatie VPOKEt op de oude positie vlak voor de regel staat die een nieuw figuurtje VPOKEt. Deze methode vermindert het geflikker van de figuren waardoor het beeld veel duidelijker en minder vermoeiend wordt. De regels 420-430 voorkomen dat de monsters de doolhof verlaten. Regel 440 stuurt ons terug naar het begin van de lus.

U wilt nu vast zelf zo snel mogelijk met dit soort spelletjes aan de slag. Maar haastige spoed is zelden goed. Loop het programma eerst nog eens stap voor stap na. Het is zaak dat u goed in de gaten krijgt waar de angels en voetklemmen in zo'n programma zich kunnen bevinden.

Hoofdstuk 13

Sprites

Een van de mooiste grafische eigenschappen van de MSX zijn de sprites of sprite-patronen. Net als zelf bedachte tekens kunt u ze elke gewenste vorm geven. Zelf bedachte tekens worden binnen het schermgeheugenverplaatst door ze rechtstreeks in adressen te VPOKEn of door ze op het scherm te PRINTen. Dit betekent dat u gebonden bent aan de maximaal 960 posities op het scherm en dat ze van de ene naar de andere positie overgebracht moeten worden. Met sprites gaat dat anders: ze hoeven niet in het schermgeheugen te worden geVPOKEt. U vertelt de computer de plaats waar de sprite moet verschijnen en dat gebeurt vervolgens. Omdat de sprite niet gebonden is aan het schermheugen, kan hij vloeiende bewegingen maken over het hele scherm.

In de volgende paragrafen behandelen we de soorten sprites, de manier waarop ze kunnen worden gemaakt, bewogen en vergroot en tot slot hoe we botsingen kunnen waarnemen.

Soorten sprites

Van sprites kunt u gebruik maken in de SCREENs 1, 2 en 3. Op de MSX zijn 4 verschillende sprite-vormen mogelijk, die kunnen worden opgeroepen door een toevoeging aan de instructie SCREEN. Onderstaande tabel geeft aan welke toevoeging welk effect heeft.

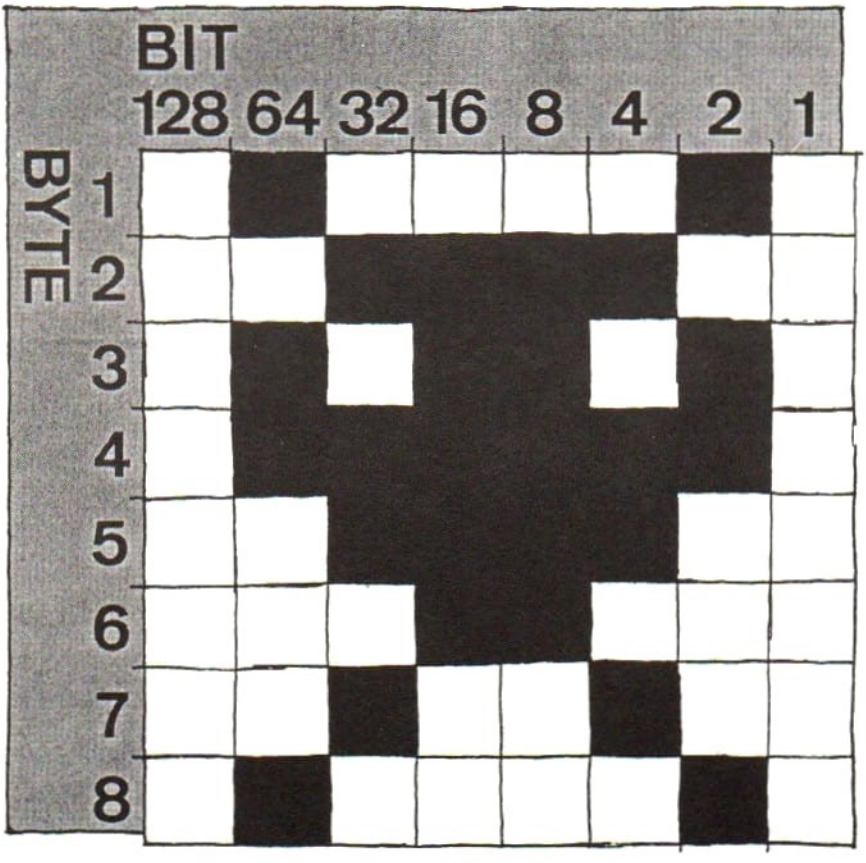
1 8 * 8 normaal
2 8 * 8 vergroot
3 16 *16 normaal
4 16 *16 vergroot

De keuze maakt u kenbaar door achter het gewenste SCREEN met een komma deze waarde toe te voegen. Zo geeft SCREEN 1,2 de sprites in SCREEN 1 vergroot weer.

Afhankelijk van het type sprite kunt u maximaal 256 verschillende sprite-patronen definiëren. Maakt u gebruik van 16*16 sprites, dan kunt u maar 64 verschillende sprite-patronen definiëren. Dat is logisch, want een 16*16 sprite neemt vier keer zoveel geheugenruimte in beslag als een 8*8 sprite. Probeert u meerdere 16*16 sprites te definiëren, dan kan dat tot vreemde effecten leiden. De computer schrijft de sprite-gegevens dan niet weg in het daarvoor bestemde gedeelte van het geheugen maar overschrijdt dit en gaat in het schermgeheugen schrijven.

Hoe definiëren we sprites

In het vorige hoofdstuk hebben we het uitgebreid gehad over het definiëren van tekens. Die kennis kunnen we prima gebruiken bij de definitie van sprites. We weten al het nodige van rasters, bits en bytes. Sprites bestaan uit een reeks karakters. De berekening van deze karakters gebeurt op dezelfde wijze als bij de herdefiniëring van karakters. We zullen die berekening nogmaals behandelen aan de hand van een voorbeeld.



We hebben weer het ruimtemannetje genomen uit hoofdstuk 10. De waarden voor de verschillende bytes hebben we nog: 66, 60, 90, 126, 60, 24, 36, 66. We maken nu een string van karakters met deze waarden, koppelen al die karakters aan elkaar en noemen het geheel SPRITE$(0). In het volgende programma geven we hiervan een voorbeeld.

```
10 REM SPRITE
20 SCREEN 1,0
30 SPRITE$(0)=CHR$(66)+CHR$(60)+CHR$(90)+CHR$
(126)+CHR$(60)+CHR$(24)+CHR$(36)+CHR$(66)
40 PUT SPRITE 0,(100,100),15,0
```

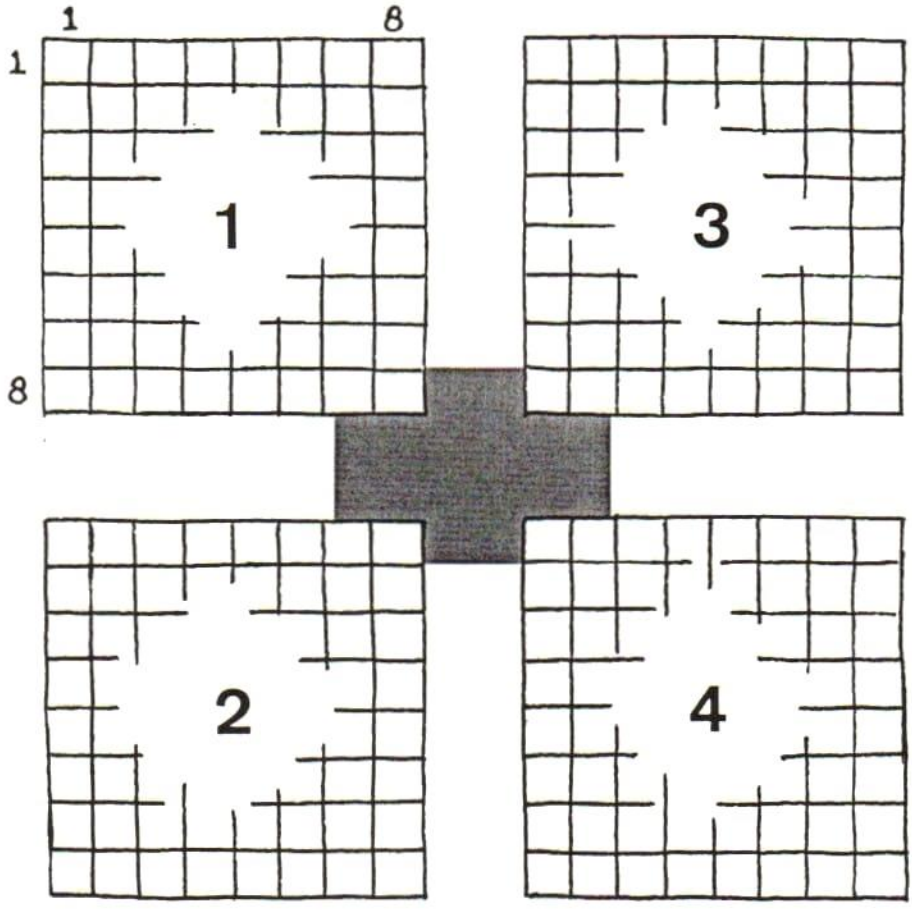
Regel 20 kiest het gewenste scherm, in dit geval SCREEN 1,0. Dat wil zeggen dat we in SCREEN 1 een normale 8*8 sprite te zien krijgen. Regel 30 telt alle karakters bij elkaar op en kent ze toe aan SPRITE$(0). Regel 40 tenslotte zet de sprite op het scherm. We laten in het volgende programma nogmaals zien hoe we een sprite kunnen definiëren, maar dan maken we gebruik van de 'grote lus' en van DATA-regels.

```
10 REM SPRITE 1
20 SCREEN 1,0
30 FOR I=0 TO 0
40 FOR B=1 TO 8
50 READ S
60 S$=S$+CHR$(S)
70 NEXT B
80 SPRITE$(I)=S$:S$=""
90 NEXT I
100 PUT SPRITE 0,(100,100),15,0
110 GOTO 110
120 DATA 66,60,90,126,60,24,36,66
```

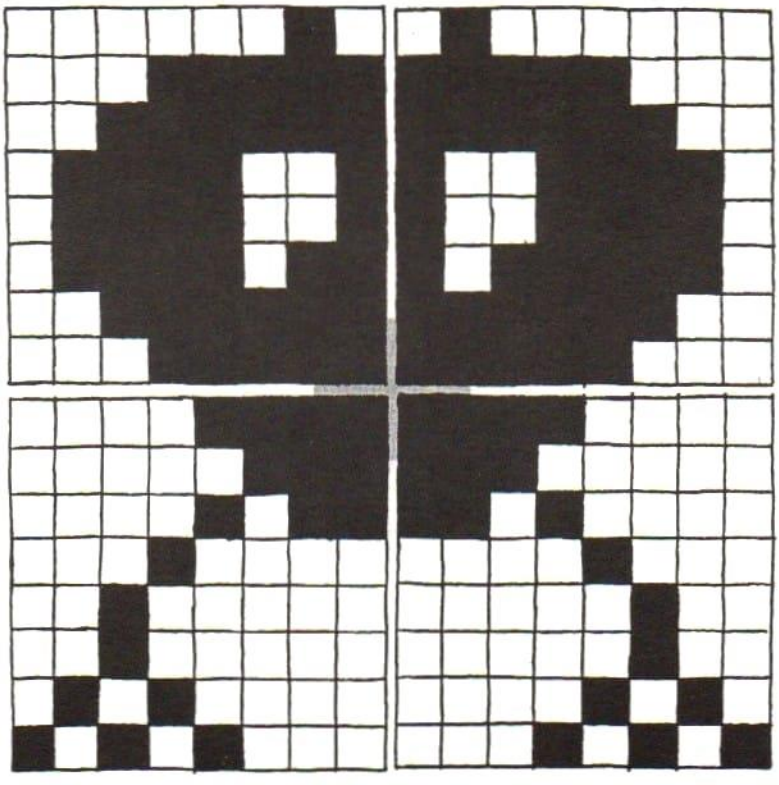
De FOR/NEXT-lus in regel 30-90 is de teller voor het aantal te definiëren sprites. In dit geval definiëren we maar één sprite, dus loopt de lus van 0 tot 0. De FOR/NEXT-lus in regel 40-70 is de teller voor het aantal karakters waaruit de sprite bestaat. Bij een 8*8 sprite zijn dat er 8, dus loopt de lus van 1 to 8. Regel 50 leest de waarden voor de karakters uit de DATA-regels. Regel 60 zet de DATA om in karakters door middel van CHR$(S) en telt ieder volgend karakter op bij het vorige. In regel 80 wordt de reeks karakters (S$) aan SPRITE$(I) toegekend en wordt de string geleegd (S$="") om voor de volgende sprite te kunnen worden gebruikt. Dan wordt de sprite met PUT SPRITE op het scherm gezet (regel 100). In de volgende paragraaf zullen we deze instructie verder behandelen. Deze routine is goed bruikbaar voor het geval u meer dan één sprite wilt gebruiken.

U zult hebben opgemerkt dat we tot nu toe alleen 8*8 sprites hebben gemaakt. Met de zojuist ontwikkelde routine is het even makkelijk om een sprite van 16*16 sprite te produceren. We verhogen daarvoor de FOR/NEXT-lus in regel 000 tot 32 en voegen extra DATA toe. In het vervolg zullen we dan ook van deze routine gebruik maken om sprites te definiëren.

In feite is een 16*16 sprite hetzelfde als vier 8*8 sprites. De volgende tekening kan dat verduidelijken.



Op de tekening ziet u dat de sprite is verdeeld in vier blokken van 8*8. De nummering van de blokken geeft aan in welke volgorde de karakters aan SPRITE$ moeten worden toegevoegd: linksboven, linksonder, rechtsboven en tot slot rechtsonder. Aan de hand van een voorbeeld zullen we dit principe toelichten.



We hebben ook nu weer een ruimtewezen als voorbeeld genomen. De verschillende blokken zijn duidelijk aangegeven en per blok zijn de waarden voor de bytes berekend. In onderstaande tabel hebben we de waarden per blok nog eens op een rijtje gezet.

blok	waarden
1	2,31,63,121,121,123,63,31
2	15,7,11,16,32,32,80,168
3	64,248,252,158,158,190,252,248
4	240,224,208,8,4,4,10,21

De DATA moeten in de gegeven volgorde aan het programma worden toegevoegd.

```
10 REM 16 X 16 SPRITE
20 SCREEN 1,2
30 FOR I=0 TO 0
40 FOR B=1 TO 32
50 READ S
60 S$=S$+CHR$(S)
70 NEXT B
80 SPRITE$(I)=S$:S$=""
90 NEXT I
100 PUT SPRITE 0,(100,100),15,0
110 GOTO 110
120 DATA 2,31,63,121,121,123,63,31
130 DATA 15,7,11,16,32,32,80,168
140 DATA 64,248,252,158,158,190,252,248
150 DATA 240,224,208,8,4,4,10,21
```

Zoals gezegd maken we weer gebruik van de tweede routine. We hebben de FOR/NEXT-lus in regel 40 verhoogd tot 32 en de extra DATA toegevoegd. SCREEN 1,0 hebben we vervangen door SCREEN 1,2 omdat we in dit geval een normale 16*16 sprite gebruiken. Regel 100 zet de sprite weer op het beeldscherm.

PUT SPRITE

Om uw sprite op het scherm te plaatsen staat u de instructie PUT SPRITE ter beschikking. De opbouw van PUT SPRITE is:

PUT SPRITE ‹spritevlak›, (X-positie,Y-positie), ‹kleur›, ‹spritenummer›

We zullen de verschillende parameters van PUT SPRITE afzonderlijk behandelen. Daarna komen in een kort programma de verschillende parameters van PUT SPRITE aan bod.

Spritevlakken

De MSX heeft 32 zogenaamde spritevlakken. Die spritevlakken moet u zich voorstellen als lagen die zich voor het tekstscherm bevinden. Elk vlak bevat de X- en Y-postitie van 1 sprite. U kunt dus 32 sprites tegelijk op uw scherm laten verschijnen. Probeert u meer dan één sprite per vlak af te beelden, dan zal de laatst afgebeelde sprite zicht-

baar worden. De voorlaatste wordt vervangen door de laatste. Het spritevlak geeft verder de prioriteit van de sprite aan. Hoe lager het nummer van het vlak, des te hoger de prioriteit van de sprite. Dat wil zeggen dat wanneer u probeert twee sprites in verschillende spritevlakken op dezelfde positie te zetten, de sprite met het laagste vlaknummer verschijnt vóór de sprite met het hogere vlaknummer.

X-positie en Y-positie

Deze parameters geven de positie van de sprite op het scherm aan. Het scherm is opgebouw uit 256 X-posities en 192 Y-posities. 0,0 is de positie linksboven op het scherm, 255,191 is de positie rechtsonder. Deze X- en Y-posities zijn straks van belang bij de beweging van de sprites. Opgemerkt moet nog worden dat op de horizontale lijn maximaal 4 sprites geplaatst kunnen worden. Komen er meer op één lijn, dan zullen de sprites met de laagste prioriteit niet zichtbaar zijn. Voor de Y-positie geldt geen beperking wat het aantal sprites betreft.

Kleur

Deze parameter geeft de kleur van de sprite aan. U hebt de keuze uit de 15 kleuren van de MSX. Welke waarde de verschillende kleuren hebben kunt in bijlage 2 achterin het boek lezen.

Spritenummer

Hiermee geeft u de gewenste sprite aan. Wanneer u gebruik maakt van 8*8 sprites, is dat een getal tussen de 0 en 255 en bij sprites van 16*16 een getal tussen de 0 en 63.

```
10 REM PUT SPRITE 1
20 SCREEN 1,0
30 FOR I=0 TO 1
40 FOR B=1 TO 8
50 READ S
60 S$=S$+CHR$(S)
70 NEXT B
80 SPRITE$(I)=S$:S$=""
90 NEXT I
100 FOR X1=255 TO 0 STEP -1
```

```
110 X2=255-X1
120 PUT SPRITE 0,(X1,96),15,0
130 PUT SPRITE 1,(X2,96),1,1
140 FOR TD=1 TO 5:NEXT TD
150 NEXT X1
160 DATA 16,48,112,255,255,112,48,16
170 DATA 8,12,14,255,255,14,12,8
```

Dit programma geeft twee pijlen weer, waarvan de één van links naar rechts en de ander van rechts naar links over het scherm beweegt. Daarbij kunt u zien dat de witte pijl een hogere prioriteit heeft dan de zwarte: hij gaat voor de zwarte langs. De beweging wordt verzorgd door de FOR/NEXT-lus in regel 100-160. We gebruiken de variabelen X1 en X2, waarbij X2 door X1 wordt bepaald. De snelheid van de pijlen is afhankelijk van de vertragingslus in regel 140 en kan hoger of lager worden gemaakt.

Het volgende PUT SPRITE-programmaatje laat zien dat dezelfde sprite in verschillende spritevlakken meer dan één keer op het scherm te zien kan zijn.

```
10 REM PUT SPRITE 2
20 SCREEN 1,0
30 FOR I=0 TO 0
40 FOR B=1 TO 8
50 READ S
60 S$=S$+CHR$(S)
70 NEXT B
80 SPRITE$(I)=S$:S$=""
90 NEXT I
100 L=RND(-TIME)
110 FOR S=0 TO 31
120 X=INT(RND(1)*255)+1:Y=INT(RND(1)*192)+1
130 K=INT(RND(1)*15+1)
140 PUT SPRITE S,(X,Y),K,0
150 NEXT S
160 DATA 60,126,129,153,255,36,66,129
```

In dit programma worden de spritevlakken 0 tot en met 31 stuk voor stuk in een FOR/NEXT-lus aangeroepen. De X- en Y-positie en de kleur worden at random bepaald (met RND). In regel 140 zijn alle getallen behalve het spritenummer (0) door variabelen vervangen en afhankelijk van die variabelen wordt de sprite op het scherm gezet.

In beweging

We kunnen nu sprites maken, op de gewenste plaats op het scherm zetten, een kleur geven en we weten welke sprite de hoogste prioriteit heeft. We kunnen ze ook al laten bewegen; het wordt dus de hoogste tijd om de sprites te gaan besturen. Bij de behandeling van de verschillende parameters van PUT SPRITE hebben we al aangegeven waar het bij het sprite-beweging om draait: de X- en de Y-positie van de sprite. In het volgende programma hebben we deze twee posities vervangen door variabelen. Afhankelijk van de invoer via de joystick verhogen of verlagen we deze variabelen. Daarmee is de besturing een feit!

```
10 REM PUT SPRITE 3
20 SCREEN 3,2
30 FOR I=0 TO 0
40 FOR B=1 TO 32
50 READ S
60 S$=S$+CHR$(S)
70 NEXT B
80 SPRITE$(I)=S$:S$=""
90 NEXT I
100 X=128:Y=96:K=1
110 R=STICK(1)
120 V=STRIG(1)
130 IF R=1 THEN Y=Y-1
140 IF R=3 THEN X=X+1
150 IF R=5 THEN Y=Y+1
160 IF R=7 THEN X=X-1
170 IF V<>0 THEN K=K+1:IF K=16 THEN K=1
180 PUT SPRITE 0,(X,Y),K,0
190 GOTO 110
200 DATA 0,0,0,0,0,255,255,255
210 DATA 255,255,255,0,0,0,0,0
220 DATA 0,64,96,112,120,252,254,255
230 DATA 255,254,252,120,112,96,64,0
```

De routine in regel 30 tot regel 90 is inmiddels bekend. Hier definiëren we de sprite, dit keer een pijl (rooster: 16*16). De regels 110-190 lezen met behulp van de instructie STICK(1) en STRIG(1) de joystick uit en verhogen of verlagen, afhankelijk van de invoer, de variabelen X en Y. STRIG(1) wordt in dit geval gebruikt om de kleur van de sprite te veranderen. Bij iedere druk op de vuurknop wordt de variabele K (kleur) met 1 verhoogd. Regel 170 plaatst de sprite op het scherm.

De wonderbaarlijke uitdijende sprite

Zoals reeds opgemerkt kunnen we sprites op vier verschillende manieren op het scherm zetten. Tot nu toe hebben we slechts twee mogelijkheden behandeld, de 8*8 sprite en de 16*16 sprite. Door aan SCREEN met een komma een 0, 1, 2 of 3 toe te voegen kunt u de verschillende spritegroottes zien.
Het volgende programma laat de 8*8 sprite normaal en vergroot zien en het daarop volgende programma doet hetzelfde voor de 16*16 sprite. De vergroting is afhankelijk van de toevoeging aan de instructie SCREEN. Welke toevoeging welk effect heeft kunt u lezen in de paragraaf over de verschillende sprite-types.

```
10 REM 8 X 8 SPRITE NORMAAL EN VERGROOT
20 IF X=1 THEN X=0 ELSE X=1
30 SCREEN 1,X
40 RESTORE
50 FOR I=0 TO 0
60 FOR B=1 TO 8
70 READ S
80 S$=S$+CHR$(S)
90 NEXT B
100 SPRITE$(I)=S$:S$=""
110 NEXT I
120 PUT SPRITE 0,(112,96),1,0
130 FOR TD=1 TO 1000:NEXT TD
140 GOTO 20
150 DATA 85,85,85,85,85,85,85,85
```

```
10 REM 16 X 16 SPRITE NORMAAL EN VERGROOT
20 IF X=2 THEN X=3 ELSE X=2
30 SCREEN 1,X
40 RESTORE
50 FOR I=0 TO 0
60 FOR B=1 TO 32
70 READ S
80 S$=S$+CHR$(S)
90 NEXT B
100 SPRITE$(I)=S$:S$=""
110 NEXT I
120 PUT SPRITE 0,(112,96),1,0
130 FOR TD=1 TO 1000:NEXT TD
140 GOTO 20
150 DATA 85,170,85,170,85,170,85,170
160 DATA 85,170,85,170,85,170,85,170
170 DATA 85,170,85,170,85,170,85,170
180 DATA 85,170,85,170,85,170,85,170
```

Een botsend geheel

Voor spelletjes is het vaak heel belangrijk dat we kunnen waarnemen wanneer twee verschillende sprites tegen elkaar botsen. De MSX heeft daar een speciale instructie voor: ON SPRITE GOSUB. Deze instructie werkt samen met SPRITE ON en SPRITE OFF. Aan het begin van een programma voert u de instructie ON SPRITE GOSUB in. Achter GOSUB vermeldt u het regelnummer waar het programma in geval van een botsing heen moet springen. U kunt de instructie activeren door middel van SPRITE ON, SPRITE OFF stelt de instructie juist buiten werking. In het volgende programma zijn deze beide instructies verwerkt. We laten twee sprites tegen elkaar botsen. Op het moment van de botsing zorgen we ervoor dat er een melding op het scherm verschijnt. Voor dit voorbeeld gebruiken we de beide pijltjes als sprites.

```
10 REM ON SPRITE GOSUB
20 ON SPRITE GOSUB 190
30 SPRITE ON
40 SCREEN 1,0
50 FOR I=0 TO 1
60 FOR B=1 TO 8
70 READ S
80 S$=S$+CHR$(S)
90 NEXT B
```

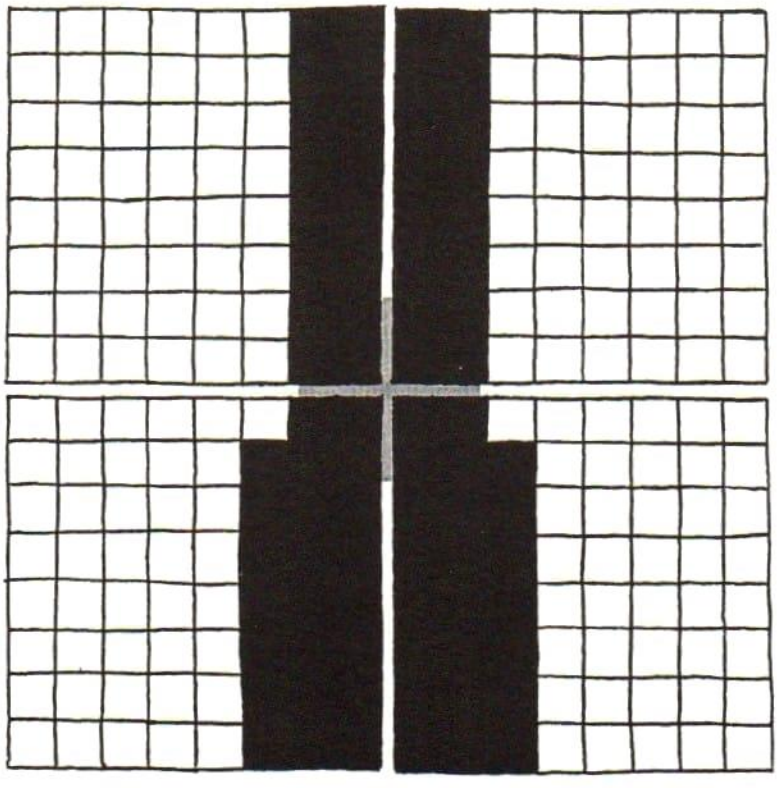
```
100 SPRITE$(I)=S$:S$=""
110 NEXT I
120 FOR X=10 TO 240
130 PUT SPRITE 1,(X,96),15,1
140 PUT SPRITE 0,(250-X,96),1,0
150 FOR TD=1 TO 5:NEXT TD
160 NEXT X
170 SPRITE ON:GOTO 120
180 END
190 SPRITE OFF
200 PRINT "KNAL"
210 BEEP:RETURN
220 DATA 16,48,112,255,255,112,48,16
230 DATA 8,12,14,255,255,14,12,8
```

In regel 20 wordt in de instructie ON SPRITE GOSUB aangegeven dat bij een botsing naar regel 190 moet worden gesprongen. Regel 30 stelt door middel van SPRITE ON de ON SPRITE GOSUB-instructie in werking. De regels 50-110 definiëren de sprite. De FOR/NEXT-lus in de regels 120-160 laat de twee sprites bewegen en tegen elkaar botsen. De subroutine in de regels 190-200 stelt de instructie ON SPRITE GOSUB buiten werking en PRINT de melding op het scherm dat er een botsing heeft plaatsgevonden.

Hoofdstuk 14

Meerkleuren-sprites

De MSX biedt niet mogelijkheid meerkleuren-sprites te maken, maar we kunnen ze wel simuleren. Door twee of meer sprites van ongeveer dezelfde vorm in verschillende spritevlakken op dezelfde X- en Y-posities te zetten en ze verschillende kleuren te geven, kunnen we een meerkleuren-sprite nabootsen. In principe zouden we op deze wijze een vierkleurensprite kunnen maken, omdat we per Y-positie maximaal 4 sprites kwijt kunnen. Maar bij deze vorm van meerkleurensprites maken we meestal gebruik van twee kleuren, omdat de verschillende sprites na elkaar op het scherm geplaatst worden. Zouden we vier sprites nemen, dan is de kans groot dat het te duidelijk zichtbaar wordt dat het niet om één maar om meerdere sprites gaat. In de volgende twee rasters hebben we twee sprites ontworpen die afzonderlijk niet helemaal af zijn, maar die, wanneer we ze voor elkaar houden, samen een race-auto vormen.



In de tabellen hieronder staan de DATA die we op de vertrouwde wijze hebben berekend.

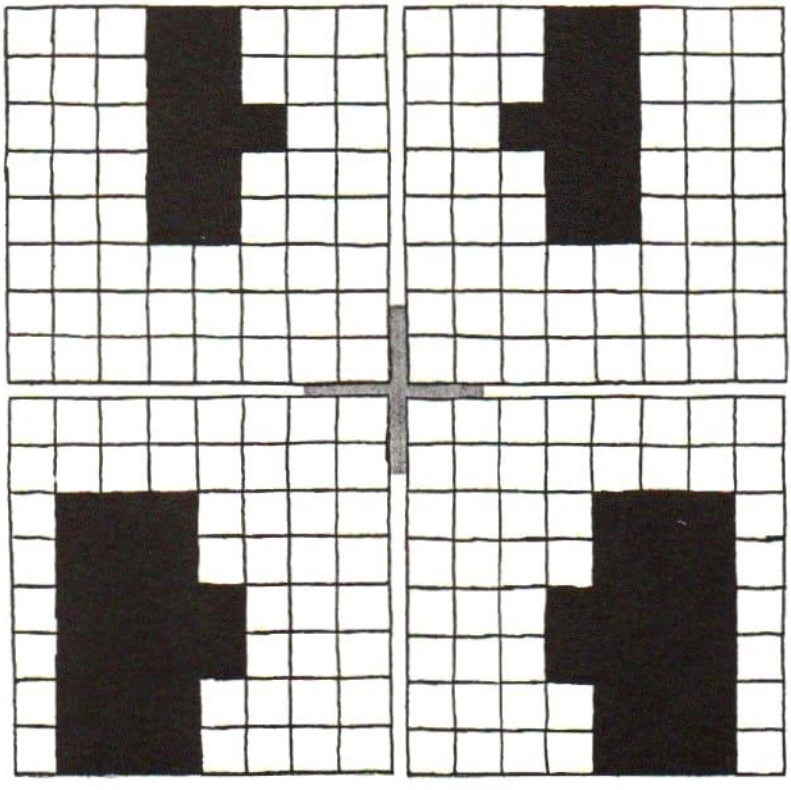
SPRITE 1

blok 1	3,3,3,3,3,3,3,3
blok 2	3,7,7,7,7,7,7,7
blok 3	192,192,192,192,192,192,192,192
blok 4	192,224,224,224,224,224,224,224

SPRITE 2

blok 1	24,24,28,24,24,0,0,0
blok 2	0,0,112,112,120,120,112,112
blok 3	24,24,56,24,24,0,0,0
blok 4	0,0,14,14,30,30,14,14

In het volgende programma laten we de race-auto in volle glorie zien.



```
10 REM MEERKLEURENSPRITE
20 SCREEN 1,3
30 FOR I=0 TO 1
40 FOR B=1 TO 32
50 READ S
60 S$=S$+CHR$(S)
70 NEXT B
80 SPRITE$(I)=S$:S$=""
90 NEXT I
100 PUT SPRITE 1,(100,100),15,0
110 PUT SPRITE 0,(100,100),1,1
120 GOTO 120
130 DATA 3,3,3,3,3,3,3,3
140 DATA 3,7,7,7,7,7,7,7
150 DATA 192,192,192,192,192,192,192,192
160 DATA 192,224,224,224,224,224,224,224
170 DATA 24,24,28,24,24,0,0,0
```

```
180 DATA 0,0,112,112,120,120,112,112
190 DATA 24,24,56,24,24,0,0,0
200 DATA 0,0,14,14,30,30,14,14
```

Hoofdstuk 15

Sprite-animatie

Met sprites kunnen we echt bewegende beelden creëren. Door een aantal momentopnames (sprites) achter elkaar te plakken die weinig van elkaar verschillen, krijgt u een mooie vloeiende beweging. Om dit principe te demonstreren hebben we een programma geschreven dat gebruikt maakt van drie eenvoudige sprites om een lopend mannetje na te bootsen. Het programma gebruikt maar één spritevlak, vlak 0. Door de waarde van het spritenummer te veranderen wijzigt het programma de vorm van het mannetje tijdens een lus. Het grootste gedeelte van het programma kent u ondertussen: de definiëring van de sprite en de manier van bewegen.

```
10 REM BEWEGENDE SPRITES
20 SCREEN 1,1
30 FOR I=0 TO 2
40 FOR B=1 TO 8
50 READ S
60 S$=S$+CHR$(S)
70 NEXT B
80 SPRITE$(I)=S$:S$=""
90 NEXT I
100 X=250:S=0
110 PUT SPRITE 0,(X,100),1,S
120 X=X-4:S=S+1:IF S=3 THEN S=0
130 FOR TD=1 TO 50:NEXT TD
140 IF X<-15 THEN END
150 GOTO 110
160 DATA 48,16,80,124,20,112,40,4
170 DATA 48,16,80,124,20,112,76,64
180 DATA 48,16,56,56,16,16,24,16
```

Na RUN staat er een eenvoudige tekening van een mannetje rechts op het scherm. Als het programma de lus in regel 110-150 doorloopt, wandelt het mannetje over het scherm. Door X steeds met 4 te verlagen wordt de sprite steeds verder naar links op het scherm gezet. De variabele S wordt gebruikt om de verschillende sprites weer te geven. S kan 0, 1 of 2 zijn; daar zorgt de routine in regel 120 voor. Ook wordt in deze regel X steeds met 4 verlaagd.

Eerder in het boek hebben we deze methode ook al toegepast. Het verschil is dat we toen geen sprites gebruikten, maar opnieuw gedefinieerde karakters. In het spel HAPPY HAPPER werd afhankelijk van de invoer van de joystick een Happer afgebeeld die in de juiste richting keek. Op welke wijze deze manier van animatie verder kan worden benut, wordt in de drie spelen aan het eind van het boek duidelijk gemaakt.

Hoofdstuk 16

Pret met sprites

Na deze zware kost wordt het tijd voor een adempauze en wat verpozing. Het programma AKK AKK van William Leader is een goed voorbeeld van de prachtige effecten die met sprites kunnen worden bereikt. Het spel maakt gebruik van vergrote sprites, die de vorm hebben van bommenwerpers. Deze bommenwerpers vallen 's nachts uw stad aan. Het afweergeschut dat de stad moet beschermen, bestaat uit een antiek kanon. De bommenwerpers zijn zwart geverfd en daardoor 's nachts onmogelijk waar te nemen aan het zwerk. De enige aanwijzing is dat ze tijdens hun vlucht sterren aan het oog onttrekken. Het enige lichtpuntje dat u heeft, is een oud zoeklicht dat de hemel afspeurt. Als u geluk hebt, vangt het een bommenwerper in zijn lichtbundel. Richt de loop omhoog en als u een vliegtuig in het vizier denkt te hebben, aarzel dan niet maar schiet. Zoals we al zeiden is het een kanon uit grootmoeders tijd en duurt het lang eer het opnieuw is geladen. U krijgt doorgaans per bommenwerper maar één kans om te vuren.

De vliegtuigen vliegen van rechts naar links. Als een vliegtuig tweederde van zijn weg heeft afgelegd zonder te zijn geraakt, laat de piloot een bom op de stad vallen. Als vijf bommenwerpers hun bom hebben laten vallen, hebt u verloren. Het gaat uiteraard om een zo hoog mogelijke score. Het spel is in het begin vrij moeilijk, maar na enige oefening raakt u al gauw bedreven in het opsporen van de bommenwerpers. Het spel heeft drie niveaus en twee schijnwerperposities. Bij elk hoger niveau vliegen de bommenwerpers sneller en zijn er minder sterren aan de hemel.



Nadat de titelpagina is gewist, kiest u een moeilijkheidsgraad. Niveau 1 is het gemakkelijkst en niveau 3 het moeilijkst. Wanneer u dit hebt gedaan, vraagt de computer welke schijnwerperpositie u wenst. Het scherm wordt vervolgens gewist, waarna de sterren overal op het scherm worden geVPOKEt. Daarna verschijnen het vizier en de gele lichtbundel van de schijnwerper op het scherm en begint het spel. Houd de rechterkant van het scherm nauwlettend in de gaten. Na een tijdje ontwikkelt u een zesde zintuig voor naderende bommenwerpers.

```
10 REM *** AKK AKK ***
20 GOSUB 1420
30 GOSUB 1610
40 GOSUB 1730
50 GOSUB 1920
60 REM *** HOOFDROUTINE ***
70 R=STICK(1)
80 V=STRIG(1)
90 IF R=1 THEN Y=Y-5:IF Y<30 THEN Y=30:REM OM
HOOG
100 IF R=3 THEN X=X+5:IF X>250 THEN X=250:REM
```

```
 RECHTS
110 IF R=5 THEN Y=Y+5:IF Y>180 THEN Y=180:REM
 OMLAAG
120 IF R=7 THEN X=X-5:IF X<0 THEN X=0:REM LIN
KS
130 REM *** DE GRENS OVERGAAN ***
140 IF X>255 THEN CB=100:SH=0
150 IF X<0 AND CB=100 THEN CB=0:SH=0
160 IF SH=0 THEN PUT SPRITE 3,(O,195),9,3
170 IF SH=100 THEN GOSUB 350:GOTO 190
180 IF V<>0 THEN O=X+5:P=Y+5:SH=100:GOSUB 350
:GOSUB 750
190 GOSUB 230
200 GOSUB 470
210 PUT SPRITE 0,(X,Y),15,0
220 GOTO 60
230 REM *** ZOEKLICHT ***
240 IF CTR=1 THEN X2=140:GOTO 320 ELSE X2=90
250 W=INT(20*RND(1)+1):IF W=20 THEN D=-D
260 X2=X2+D/3
270 IF X2<20 THEN D=-D:X2=20
280 IF X2>250 THEN D=-D:X2=250
290 IF Y2<Y3 THEN Y2=Y2+(4-LV):IF Y2>170 THEN
 Y2=170
300 IF Y2>Y3 THEN Y2=Y2-(4-LV):IF Y2<20 THEN
Y2=20
310 PUT SPRITE 2,(X2,Y2),11,2:RETURN
320 IF X2<X3 THEN X2=X2+(4-LV):IF X2>250 THEN
 X2=250
330 IF X2>X3 THEN X2=X2-(4-LV):IF X2<20 THEN
X2=20
340 GOTO 280
350 REM *** SCHOT ***
360 PUT SPRITE 3,(O,P),9,3
370 IF SH=105 THEN GOSUB 400
380 IF SH=108 THEN GOSUB 400
390 SH=SH+1:GOSUB 400:RETURN
400 IF O<X3+15 AND O>X3-15 AND P<Y3+15 AND P>
Y3-15 THEN SC=SC+100:GOTO 430
410 SH=0
420 RETURN
430 GOSUB 840
440 PUT SPRITE 1,(100,195),15,1
450 PUT SPRITE 3,(100,195),15,3
```

```
460 FL=0:X3=0:Y3=0:RETURN
470 REM *** VLIEGTUIG ***
480 IF FL=100 THEN 520
490 T=INT(10*RND(1)+1)
500 IF T=10 THEN GOSUB 720
510 RETURN
520 X3=X3-(4+(LV)):Y3=Y3+INT(5*RND(1)-2):IF X
3<0 AND AB=100 THEN GOTO 590
530 IF X3<65 AND X3>50 THEN 610
540 IF Y3<30 THEN Y3=30
550 IF Y3>160 THEN Y3=160
560 IF X3<0 THEN X3=0:Y3=195:FL=0
570 PUT SPRITE 1,(X3,Y3),1,1:RETURN
580 REM *** VLIEGTUIG DOOR BARRIERE ***
590 PUT SPRITE 1,(255,Y3),8,1:AB=0:X3=255:GOT
O 570
600 REM *** BOMBARDEMENT ***
610 GOSUB 960
620 FOR T=1 TO 3
630 COLOR 1,10,1
640 FOR U=1 TO 100:NEXT U
650 COLOR 10,1,10
660 FOR U=1 TO 100:NEXT U
670 NEXT T
680 COLOR 15,1,1
690 BM=BM+1:IF BM=5 THEN 1230
700 X3=X3-15:GOTO 570
710 REM *** VLIEGTUIG INITIALISEREN ***
720 FL=100:AB=100
730 X3=250:Y3=INT(180*RND(1)+1)
740 RETURN
750 REM *** GELUID FLAK ***
760 SOUND 0,20
770 SOUND 1,0
780 SOUND 6,20
790 SOUND 7,54
800 SOUND 8,15
810 FOR TD=1 TO 500:NEXT TD
820 SOUND 8,0
830 RETURN
840 REM *** EXPLOSIEGELUID ***
850 SOUND 0,250
860 SOUND 1,2
870 SOUND 6,28
```

```
880 SOUND 7,55
890 SOUND 8,16
900 SOUND 11,1
910 SOUND 12,255
920 SOUND 13,0
930 FOR TD=1 TO 1000:NEXT TD
940 RETURN
950 REM *** GELUID BOM ***
960 SOUND 0,250
970 SOUND 1,5
980 SOUND 7,62
990 SOUND 8,16
1000 SOUND 11,1
1010 SOUND 12,255
1020 SOUND 13,1
1030 FOR TD=1 TO 500:NEXT TD
1040 RETURN
1050 REM *** EINDMUZIEK ***
1060 RESTORE 1150:FOR D=1 TO 39
1070 READ N
1080 READ L
1090 PLAY "L=L;N=N;"
1100 A$=INKEY$:IF A$<>"" THEN GOSUB 1420:GOTO
 40
1110 NEXT D
1120 A$=INKEY$:IF A$="" THEN 1120
1130 GOSUB 1420
1140 GOTO 40
1150 DATA 52,6,55,3,57,6,59,4,60,12,59,6
1160 DATA 57,3,53,6,50,4,52,12,53,6
1170 DATA 55,3,52,6,52,4,51,12,52,6,53,3
1180 DATA 50,6,47,3,52,6
1190 DATA 55,3,57,6,59,4,60,12,59,6
1200 DATA 57,3,53,6,50,4,52,12,53,6
1210 DATA 55,4,53,12,52,6,50,4,49,12
1220 DATA 50,6,52,3,52,6,52,2
1230 REM EINDSEQUENTIE
1240 COLOR 15,4,4
1250 SCREEN 1
1260 CLS
1270 LOCATE 2,10
1280 PRINT"AAAAAAAAAAAARRRGGGHHHHH!!!"
1290 FOR TD=1 TO 3000:NEXT TD
1300 LOCATE 2,12
```

```
1310 PRINT"DE STAD IS VERWOEST"
1320 FOR TD=1 TO 1000:NEXT TD
1330 LOCATE 2,14
1340 PRINT"DE SCORE IS ";SC;" PUNTEN"
1350 LOCATE 2,16
1360 PRINT"DE POGING OM DE STAD TE"
1370 LOCATE 2,18
1380 PRINT"REDDEN IS MISLUKT"
1390 LOCATE 8,22
1400 PRINT"DRUK EEN TOETS"
1410 GOTO 1050
1420 REM *** INITIALISERING ***
1430 CLS
1440 KEY OFF
1450 SCREEN 1,3
1460 K=8192
1470 Q=RND(-TIME)
1480 D=1:X2=100:Y2=100:BM=0
1490 X=128:Y=96:AB=0:SC=0
1500 LV=VAL(LV$):CTR=VAL(CTR$)
1510 REM SPRITES DEFINIEREN
1520 RESTORE 2100
1530 FOR I=0 TO 3
1540 FOR B=1 TO 32
1550 READ S
1560 S$=S$+CHR$(S)
1570 NEXT B
1580 SPRITE$(I)=S$:S$=""
1590 NEXT I
1600 RETURN
1610 REM *** TITELPAGINA ***
1620 COLOR 1,14,14
1630 RESTORE 2310
1640 VPOKE K+5,206
1650 FOR Q=1 TO 15
1660 READ B
1670 LOCATE 5+Q,12,0
1680 PRINT CHR$(B)
1690 FOR TD=1 TO 100:NEXT TD
1700 NEXT Q
1710 FOR TD=1 TO 2000:NEXT TD
1720 RETURN
1730 REM MOEILIJKHEIDSGRAAD
1740 CLS
```

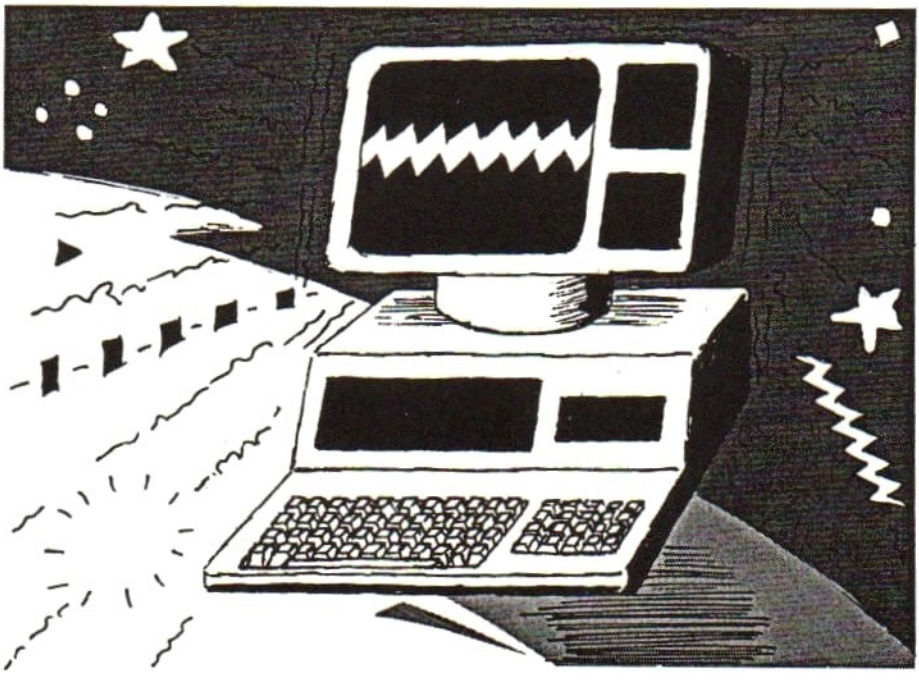
```
1750 LOCATE 2,12
1760 PRINT"WELKE MOEILIJKHEIDSGRAAD"
1770 LOCATE 2,13
1780 PRINT"WILT U: 1 IS HET GEMAK-"
1790 LOCATE 2,14
1800 PRINT"KELIJKST EN 3 IS HET MOEI-"
1810 LOCATE 2,15
1820 PRINT"LIJKST"
1830 LV$=INKEY$
1840 IF LV$<>"1" AND LV$<>"2" AND LV$<>"3" TH
EN 1830
1850 LOCATE 2,17
1860 PRINT"WELKE SCHIJNWERPERPOSITIE"
1870 LOCATE 2,18
1880 PRINT"1 IS GEMAKKELIJK, 2 MOELIJK"
1890 CTR$=INKEY$
1900 IF CTR$<>"1" AND CTR$<>"2" THEN 1890
1910 RETURN
1920 REM *** HET SCHERM KLAARMAKEN ***
1930 CLS
1940 COLOR 15,1,1
1950 LV=VAL(LV$)
1960 CTR=VAL(CTR$)
1970 COLOR 15,1,1
1980 ON LV GOTO 1990,2010,2030
1990 ST=21
2000 GOTO 2050
2010 ST=14
2020 GOTO 2050
2030 ST=7
2040 GOTO 2050
2050 FOR Z=1 TO ST
2060 E=INT(RND(1)*767+1)
2070 VPOKE 6144+E,46
2080 NEXT Z
2090 RETURN
2100 REM *** DATA VOOR SPRITES ***
2110 REM *** DATA KANONSLOOP ***
2120 DATA 0,0,3,5,9,17,35,63
2130 DATA 63,35,17,9,5,3,0,0
2140 DATA 0,0,192,160,144,136,196,252
2150 DATA 252,196,136,144,160,192,0,0
2160 REM *** DATA VOOR VLIEGTUIG ***
2170 DATA 3,7,7,7,135,135,191,255
```

```
2180 DATA 255,191,135,135,7,7,7,3
2190 DATA 0,128,128,129,129,129,163,255
2200 DATA 255,163,129,129,129,128,128,0
2210 REM *** DATA VOOR ZOEKLICHT ***
2220 DATA 7,15,31,63,127,255,255,255
2230 DATA 255,255,255,127,63,31,15,7
2240 DATA 224,240,248,252,254,255,255,255
2250 DATA 255,255,255,254,252,248,240,224
2260 REM *** DATA VOOR EXPLOSIE ***
2270 DATA 63,108,75,247,97,140,43,76
2280 DATA 12,212,165,62,30,236,185,18
2290 DATA 104,190,235,155,171,45,36
2300 DATA 126,220,122,54,179,70,21
2310 REM *** DATA VOOR TITELPAGINA ***
2320 DATA 42,42,42,32,65,75,75,32
2330 DATA 65,75,75,32,42,42,42
```

Hoofdstuk 17

Greep op geluid: SOUND

De geluids-chip van de MSX is aan het eind van de 70-er jaren ontwikkeld. Het gaat om de PSG, de Programmable Sound Generator chip. Misschien denkt u dat zo'n chip in de tussentijd alweer verouderd is, maar het is nog altijd één van de meest gebruikte chips voor computers als de MSX. De reden dat deze chip nog altijd zeer gewild is, is dat hij in staat is om over 3 verschillende kanalen tegelijkertijd geluid en witte ruis te produceren. Een ander voordeel van deze chip is dat het een processor is die geheel op zichzelf staat en geen gebruik maakt van de hoofdprocessor. Dus ondanks het feit dat de chip niet de meest geavanceerde is, kunnen we er een hoop mee doen.



De MSX heeft twee manieren waarop geluid kan worden gegenereerd: met behulp van de instructie SOUND en met behulp van de instructie PLAY. In dit hoofdstuk behandelen we de instructie SOUND. We laten aan de hand van de 14 verschillende sound-registers zien hoe u van uw computer een 'radio lawaaipapegaai' kunt maken. In hoofdstuk 17 laten we de computer deuntjes spelen met behulp van de instructie PLAY.

Veertien verschillende registers

We zullen alle veertien registers afzonderlijk behandelen, tenzij meerdere registers dezelfde functie hebben voor de verschillende kanalen. Voor elk van de veertien registers geldt de volgende syntaxis:

SOUND ‹register›,‹waarde›

De opbouw van deze paragraaf is als volgt: we geven aan om welk register het gaat, wat de betekenis van het register is en in welk bereik de waarde moet liggen. Daarna vertellen we iets meer over het betreffende register.

Fijne instelling voor kanaal A, B en C: registers 0, 2 en 4

bereik: 0-255

Deze drie registers zorgen voor de fijnafstemming van de kanalen A, B en C. Deze registers hangen nauw samen met de volgende drie, die de grove geluidsinstelling van de kanalen A, B en C regelen.

Grove instelling voor kanaal A, B en C: registers 1, 3 en 5

bereik: 0-15

Zoals gezegd zorgen deze drie kanalen voor de grove instelling van het geluid per kanaal. U kunt zich dat zo voorstellen: met deze registers kunt u 15 verschillende toonhoogten genereren en met de registers voor de fijnafstemming kunt u tussen elke twee grove tonen 255 andere tonen genereren.

Het volgende programma verduidelijkt een en ander.

```
10 REM SOUND 1
20 CLS
30 LOCATE 6,10:PRINT"FIJN:"
40 LOCATE 6,11:PRINT"GROF:"
50 FOR Y=0 TO 15
60 FOR X=0 TO 255 STEP 5
70 LOCATE 10,10:PRINT X
80 LOCATE 10,11:PRINT Y
90 SOUND 0,X
100 SOUND 1,Y
110 SOUND 7,62
120 SOUND 8,8
130 FOR TD=1 TO 10:NEXT TD
140 NEXT X
150 NEXT Y
160 STOP
```

Door middel van twee FOR/NEXT-lussen verhogen we via kanaal A stapsgewijs de grove en de fijne geluidsinstelling. In de regels 70 en 80 PRINTen we de instelling van de beide registers op dat moment.

Met de voornoemde registers kunnen we de gewenste geluidsfrequentie, uitgedrukt in Herz (Hz), op de verschillende kanalen laten horen. De waarden van de registers zijn niet in Hz ingevoerd, maar daarvoor is de volgende formule ontwikkeld:

$$\frac{1789772{,}5}{(16*Hz)} = 256*(\text{register } 1, 3, 5) + (\text{register } 0, 2, 4)$$

Wilt u nu bijvoorbeeld een frequentie van 400 Hz via kanaal A voortbrengen, dan wordt de berekening als volgt:

$$\frac{1789772{,}5}{(16*400)} = 279 = 256*1 + 23$$

Daaruit volgt dat u in register 0 de waarde 23 moet schrijven en in register 1 de waarde 1. Wilt u het geluid via kanaal B horen, dan gebruikt u in plaats van de registers 0 en 1 de registers 2 en 3. Om de omzetting van Herz naar de in te voeren waarden eenvoudiger te maken hebben we het volgende programma ontwikkeld, waarin bovenstaande formule is verwerkt.

```
10 REM HERZ
20 CLS
30 LOCATE 0,5:INPUT"HERZ-FREQUENTIE";HZ
40 F=INT(1789772.5#/(16*HZ))
50 LOCATE 0,8
60 PRINT"REGISTER 0, 2 OF 4:";F MOD 256
70 LOCATE 0,9
80 PRINT"REGISTER 1, 3 OF 5:";INT(F/256)
```

Ruisfrequentie: register 6

bereik: 0-31

De verschillende kanalen kunnen niet alleen tonen produceren maar ook de zogenaamde witte ruis. Register 6 geeft u de mogelijkheid de frequentie van deze ruis te bepalen. Hoe hoger de de waarde van register 6, hoe lager de frequentie van de ruis. Ook dit zullen we met een voorbeeld verduidelijken.

```
10 REM SOUND 2
20 CLS
30 SOUND 7,55
40 SOUND 8,15
50 FOR F=0 TO 31
60 SOUND 6,F
70 FOR TD=1 TO 200:NEXT TD
80 NEXT F
90 STOP
```

Kanaalkeuze voor het produceren van geluid: register 7

bereik: 0-63

Dit register is van zeer groot belang bij het produceren van geluid. In dit register kiest u welk kanaal u 'aanzet', en waarvoor: u kunt elk van de drie kanalen aanzetten voor ruis en of voor geluid. Elke keuze

wordt vertegenwoordigd door één bit. In onderstaande tabel wordt dit toegelicht.

bit	kanaal	waarde
7	ongebruikt	0
6	ongebruikt	0
5	C-ruis	32
4	B-ruis	16
3	A-ruis	8
2	C-geluid	4
1	B-geluid	2
0	A-geluid	1

De bits 6 en 7 zijn niet van invloed op de keuze en hebben dan ook permanent de waarde 0. Staat een kanaal uit, dan is de bit gezet. Indien alle kanalen uit staan, is de waarde van register 7 gelijk aan de waarde van bit 0 tot en met 5: 63. Wilt u een kanaal aanzetten, dan moet u de waarde van dat kanaal aftrekken van 63. Een voorbeeld: u wilt via kanaal A een geluid en via kanaal B en C een ruis horen. Voer bij register 7 het volgende in:

63 - (32 + 16 + 1) = 14

Volume: registers 8, 9 en 10

bereik: 0-16

Het volume van de kanalen A, B en C wordt respectievelijk door register 8, 9 en 10 geregeld. Het bereik van het volume ligt tussen de 0 en 15, waarbij 0 betekent dat er helemaal niets te horen is en 15 het maximale volume is. Wilt u gebruik maken van de verschillende golfvormen, dan moet u als waarde 16 invoeren.

Frequentie van de golfvormen (laag): register 11

bereik: 0-255

Met dit register kunt u de frequentie van het golfvormpatroon van de grove geluidsinstelling veranderen. De keuze van het patroon is afhankelijk van de invoer bij register 13.

Frequentie van de golfvormen (hoog): register 12

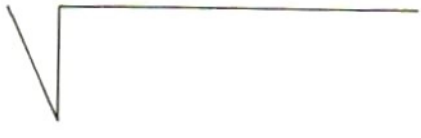
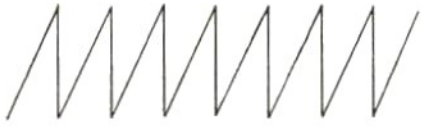
bereik: 0-255

Dit register is gelijk aan register 11, alleen kunt u met dit register de frequentie van het golfvormpatroon van de fijne geluidsinstelling veranderen. Het programma na de beschrijving van de verschillende patronen demonstreert de werking van register 11 en 12.

Keuze van het golfvormpatroon: register 13

bereik: 0-15

Met behulp van dit register kunt u 8 verschillende patronen voor golfvormen kiezen.

nummer	**patroon**
0,1,2,3,9	
4,5,6,7,15	
8	
10	
11	
12	

13



14



De tekening geeft het patroon van de golfvorm aan. Zo laat patroon 13 een aanzwellende toon horen en patroon 14 een toon die steeds aanzwelt, daarna weer afneemt, weer aanzwelt, enzovoort.

Met het volgende programma kunt u de verschillende patronen beluisteren.

```
10 REM SOUND 3
20 CLS
30 LOCATE 2,5
40 PRINT"LET OP: DE GOLFVORMEN 0,1,2,3,9"
50 LOCATE 2,6
60 PRINT"EN 4,5,6,7,15 ZIJN GELIJK"
70 SOUND 0,120
80 SOUND 1,1
90 SOUND 2,255
100 SOUND 3,0
110 SOUND 7,46
120 SOUND 8,16
130 SOUND 9,16
140 SOUND 11,10
150 SOUND 12,10
160 FOR V=0 TO 15
170 SOUND 13,V
180 LOCATE 2,8
190 PRINT"GOLFVORM:";V
200 A$=INKEY$:IF A$="" THEN 200
210 NEXT V
220 STOP
```

Regel 190 zorgt er met behulp van een FOR/NEXT-lus voor dat alle verschillende golfvormen aan bod komen. Regel 200 laat aan u de keus hoe lang u elke golfvorm wilt horen; bij elke toetsdruk begint de volgende golfvorm.

Special effects

Tot slot van dit hoofdstuk nog een paar korte programmaatjes waarin enkele *special effects* zijn verwerkt.

Tijdbom

```
10 REM TIJDBOM
20 SOUND 0,85
30 SOUND 1,0
40 SOUND 7,62
50 SOUND 8,16
60 SOUND 11,10
70 SOUND 12,10
80 SOUND 13,8
90 FOR TD=1 TO 2000:NEXT TD
100 REM EXPLOSIE
110 SOUND 0,145
120 SOUND 1,0
130 SOUND 6,15
140 SOUND 7,55
150 SOUND 11,255
160 SOUND 12,255
170 SOUND 13,0
```

Sirene

```
10 REM SIRENE
20 A=1:B=1
30 B=B+A
40 IF ABS(B)>30 THEN A=-A
50 SOUND 0,105+B
60 SOUND 1,0
70 SOUND 8,15
80 SOUND 7,62
90 GOTO 30
```

Trein

```
10 REM TREIN
20 KEY OFF
30 CLS
40 SCREEN 0
50 R=INT(RND(-TIME)*100+1)
60 IF R<>1 THEN 160
70 LOCATE 11,12
80 PRINT"TOOOOEEEETTTT!!!"
90 SOUND 0,36
100 SOUND 1,1
110 SOUND 8,14
120 SOUND 7,62
130 FOR TD=1 TO 500:NEXT TD
140 LOCATE 11,12
150 PRINT SPC(16)
160 SOUND 0,36
170 SOUND 1,2
180 SOUND 6,20
190 SOUND 7,55
200 SOUND 8,16
210 SOUND 11,3
220 SOUND 12,5
230 SOUND 13,13
240 FOR TD=1 TO 200-P:NEXT TD
250 P=P+1.5:IF P>140 THEN P=140
260 GOTO 50
```

Hoofdstuk 18

Meer geluid: PLAY

Naast de instructie SOUND kent de MSX nog een lawaaimaker: PLAY. PLAY is hetzelfde type instructie als SOUND. Het verschil zit hem in het feit dat met de instructie PLAY via een aantal deelinstructies in de verschillende registers wordt geschreven terwijl u dat met SOUND rechtstreeks doet. Daardoor is PLAY eenvoudiger te gebruiken wanneer u een deuntje ten gehore wil brengen. In dit hoofdstuk zullen we de verschillende deelinstructies van PLAY afzonderlijk behandelen. Net als in het vorige hoofdstuk ondersteunen we de theorie met korte programma's.
De instructie PLAY heeft de volgende syntaxis:

PLAY ‹string voor kanaal 1›, ‹string voor kanaal 2›, ‹string voor kanaal 3›

In die strings kunnen dan de voornoemde deelinstructies komen te staan. We zullen ze stuk voor stuk behandelen.

A, B, C, D, E, F, G

Dit zijn de noten die kunnen worden gespeeld. Door een # (matje), een plus- of een minteken toe te voegen kunt ook de (denkbeeldige) zwarte toetsen van het toetsenbord laten horen. Bedenk wel dat die toevoegingen alleen mogelijk zijn als de gevraagde noot in werkelijkheid ook bestaat. De noten worden gespeeld in het octaaf dat op dat moment is ingesteld. De volgende deelinstructie laat zien hoe u van octaaf kunt wisselen. Eerst een programmaatje dat enkele tonen laat horen.

```
10 REM PLAY 1
20 PLAY"ACAGAGB"
```

O ‹n›

bereik: 1-8

Deze deelinstructie kiest een octaaf tussen de 1 en de 8, waarbij 1 het laagste en 8 het hoogste octaaf is. Definieert u deze deelinstructie niet, dan wordt als default octaaf het vierde octaaf ingesteld. Dat is het octaaf waarin de C klinkt als de middelste C op de piano. Met behulp van een FOR/NEXT-lus laten we toon C in alle verschillende octaven horen.

```
10 REM PLAY 2
20 FOR I=1 TO 8
30 PLAY "O=I;C"
40 NEXT I
```

N ‹n›

bereik: 0-96

Bent u het octaaf-gebeuren beu, dan is deze instructie voor u de oplossing. Het biedt de mogelijkheid om binnen alle acht octaven alle noten te spelen. Noot 36 is in deze constructie de middelste C van de piano. Met deze deelinstructie is het ook mogelijk om met behulp van DATA-regels een melodie te spelen.
De volgende tabel geeft alle verschillende noten in de acht octaven aan, met erachter hun waarde:

noot/octaaf	waarde
C-1	0
C#-1	1
D-1	2
D#-1	3
E-1	4
F-1	5
F#-1	6
G-1	7
G#-1	8
A-1	9
A#-1	10
B-1	11
C-2	12
C#-2	13
D-2	14
D#-2	15
E-2	16
F-2	17
F#-2	18
G-2	19
G#-2	20
A-2	21
A#-2	22
B-2	23
C-3	24
C#-3	25
D-3	26
D#-3	27
E-3	28
F-3	29
F#-3	30
G-3	31
G#-3	32
A-3	33
A#-3	34
B-3	35
C-4	36
C#-4	37
D-4	38
D#-4	39
E-4	40
F-4	41
F#-4	42
G-4	43
G#-4	44

A-4	45
A#-4	46
B-4	47
C-5	48
C#-5	49
D-5	50
D#-5	51
E-5	52
F-5	53
F#-5	54
G-5	55
G#-5	56
A-5	57
A#-5	58
B-5	59
C-6	60
C#-6	61
D-6	62
D#-6	63
E-6	64
F-6	65
F#-6	66
G-6	67
G#-6	68
A-6	69
A#-6	70
B-6	71
C-7	72
C#-7	73
D-7	74
D#-7	75
E-7	76
F-7	77
F#-7	78
G-7	79
G#-7	80
A-7	81
A#-7	82
B-7	83
C-8	84
C#-8	85
D-8	86
D#-8	87
E-8	88
F-8	89

F#-8	90
G-8	91
G#-8	92
A-8	93
A#-8	94

Evenals bij het vorige programma hebben we gebruik gemaakt van een FOR/NEXT-lus om alle waarden van N te laten horen.

```
10 REM PLAY 3
20 FOR I=0 TO 96
30 PLAY "N=I;"
40 NEXT I
```

L ‹n›

bereik: 1-64

Met deze instructie kunt u de lengte van de noot bepalen. Wanneer u de lengte niet definieert, krijgt deze de default waarde 4. Daarbij komt L1 overeen met de lengte van een hele noot, L2 met een halve noot, L8 met een achtste, enzovoort. Behalve met L ‹n› kunt u de lengte van een noot ook veranderen door het getal van de lengte achter de betreffende noot te zetten. Zo is bijvoorbeeld L2B hetzelfde als B2. Opgemerkt moet wel dat deze contructie niet te gebruiken is bij de deelinstructie N ‹n›.

R ‹n›

bereik: 1-64

Deze instructie is in feite gelijk aan L ‹n›, met dien verstande dat het bij R ‹n› gaat om de lengte van een rust. Voert u alleen een R of R0 in dan wordt de default rust R4 (een kwart noot) als rusttijd genomen.

.
De punt: met de punt hebt u de mogelijkheid de noten A tot en met G, de noten van N ‹n› of een R met tweederde te verlengen. Het is zelfs mogelijk meer dan één punt achter een noot te plaatsen.

Tikt u eens in

PLAY "CDE"

en

PLAY "C.D.E.."

Een duidelijk verschil is hoorbaar: de eerste drie tonen klinken veel korter dan de tweede drie. De E hebben we zelfs extra verlengd door er twee punten achter te zetten.

T ‹n›

bereik: 32-255

Het tempo waarmee u de noten wil laten spelen kunt u regelen met deze deelinstructie. ‹N› is het aantal kwart noten dat per minuut gespeeld worden. Default zijn dit er 120. In het volgende programma laten we een toonladder in vier verschillende tempi horen.

```
10 REM PLAY 4
20 FOR I=50 TO 250 STEP 50
30 PLAY "T=I;O4CDEFGABO5C"
40 NEXT I
```

M ‹n›

bereik: 1-65535

Deze deelinstructie hangt nauw samen met de volgende: hij bepaalt de modulatie van de golfvorm, dat wil zeggen de snelheid waarmee de toon aanzwelt en weer afneemt. De default waarde van M is 255.

S ‹n›

bereik: 0-15

Met deze deelinstructie kunt u uit 8 verschillende golfvormen kiezen. De golfvormen zijn dezelfde als die in het vorige hoofdstuk beschreven zijn. We zullen ze dan ook niet nogmaals behandelen. De nummers van de verschillende golfvormen zijn gelijk aan die in register 13 bij de instructie SOUND.

X ‹variabele›;

Bij de instructie PLAY hebt u door middel van de deelinstructie X de mogelijkheid om noten in een string te laten horen. De string moet dan worden voorafgegaan door de X. Hebt u bijvoorbeeld A$ gedefinieerd en wilt u die spelen, dan is de syntaxis: PLAY "XA$;" Let erop

dat achter de deelinstructie een puntkomma komt te staan. In het algemeen geldt voor alle deelinstructie dat wanneer er een variabele wordt gebruikt, deze altijd moet worden gevolgd door een puntkomma.
In het volgende programma gebruiken we de deelinstructie X ‹variabele›; om dezelfde notenreeks als in het vorige programma te laten horen, in acht verschillende octaven.

```
10 REM PLAY 5
20 A$="CDEFGAB"
30 FOR I=1 TO 8
40 PLAY "O=I;XA$;"
50 NEXT I
```

De variabele A$ wordt in regel 20 gedefinieerd en in regel 40 wordt met behulp van de deelinstructie X ‹variabele›; de string gespeeld.

Meer stemmen

Tot nu toe hebben we steeds gebruikt gemaakt van het eerste kanaal. Maar uit de syntaxis van de instructie PLAY bleek dat we de mogelijkheid hebben om over drie kanalen tegelijk tonen te laten klinken. U moet dan achter PLAY de verschillende deelinstructies scheiden door komma's. Het volgende programma illustreert dat.

```
10 REM PLAY 6
20 A$="T100O2CDEFGABO3C"
30 B$="T150O4CDEFGABO5C"
40 C$="T255O6CDEFGABO7C"
50 PLAY A$
60 PLAY B$
70 PLAY C$
80 PLAY A$
90 PLAY A$,B$
100 PLAY A$,B$,C$
```

We hebben drie verschillende variabelen (A$, B$, C$) gedefinieerd, die alle drie een toonladder spelen, maar in verschillende octaven en in verschillende tempi. De regels 50-70 laten de variabelen afzonderlijk horen. Daarna laat regel 80 eerst alleen kanaal A horen, regel 90 vervolgens de kanalen A en B samen en tot slot regel 100 de kanalen A, B en C tegelijk. We maken in dit geval geen gebruik van de deelin-

structie X ‹variabele›; omdat we alle benodigde informatie in de strings hebben gezet.

Muziek

Afzonderlijke tonen en toonladders hebben we ondertussen wel genoeg gehoord. Het laatste programma van dit hoofdstuk laat een melodietje horen, waarbij we gebruik hebben gemaakt van twee deelinstructies: L voor de lengte van de noot en N voor de noot zelf. De waarden voor L en N hebben we in een aantal DATA-regels gezet. Met READ N en READ L in regel 30 en 40 lezen we de DATA.

```
10 REM DEUNTJE MET BEHULP VAN DATA
20 FOR D=1 TO 39
30 READ N
40 READ L
50 PLAY "L=L;N=N;"
60 NEXT D
1000 DATA 52,6,55,3,57,6,59,4,60,12,59,6
1010 DATA 57,3,53,6,50,4,52,12,53,6
1020 DATA 55,3,52,6,52,4,51,12,52,6,53,3
1030 DATA 50,6,47,3,52,6
1040 DATA 55,3,57,6,59,4,60,12,59,6
1050 DATA 57,3,53,6,50,4,52,12,53,6
1060 DATA 55,4,53,12,52,6,50,4,49,12
1070 DATA 50,6,52,3,52,6,52,2
```

Hoofdstuk 19

Grafiek, bit voor bit

In de vorige hoofdstukken hebben we gezien hoe we eigen tekens konden definiëren door bits te zetten in een 8*8 raster, (acht bytes onder elkaar). Met de zogenaamde *bit-mapped* grafiek kunnen we dezelfde truc uithalen voor het hele scherm. *Bit-mapped* wil zeggen dat iedere punt (pixel) op het scherm apart kan worden aangestuurd. Op de MSX kan dat in de grafische schermen 2 en 3, ook wel de *Hi-Res* modi genoemd. Hi-Res is de afkorting voor *High Resolution*, de Engelse term voor hoge oplossing, ofwel veel puntjes op het scherm. Het beeldscherm is 256 punten breed en 192 punten hoog. U kunt in deze modus meetkundige figuren tekenen of zelf afbeeldingen maken.

De MSX heeft een heel arsenaal aan instructies die het mogelijk maken om in de grafische schermen leuke effecten te bereiken. Wij zullen ons in dit boek bezighouden met drie instructies die daarmee te maken hebben. Eerst komen PSET en PRESET aan bod, twee aan elkaar verwante instructies, waarmee u één puntje op het beeldscherm kunt zetten en weer weghalen. Als derde instructie hebben we LINE gekozen, waarmee u de mogelijkheid heeft tussen twee punten een rechte lijn te trekken.

Allereerst de syntaxis van PSET en PRESET:

PSET (X-positie,Y-positie) ‹,kleur›

PRESET (X-positie,Y-positie) ‹,kleur›

Zoals u ziet is het formaat van PSET en het bijbehorende PRESET gelijk. Bij de X- en de Y-positie kunt u wiskundige formules gebruiken,

zoals we in de volgende voorbeelden zullen zien. Het eerste programma tekent een sinuscurve op een wit scherm en het tweede een tangenscurve.

```
10 REM SINUSCURVE
20 SCREEN 2
30 OPEN "GRP:" FOR OUTPUT AS #1
40 COLOR 1,15,15
50 LINE (63,96)-(188,96),1
60 LINE (63,46)-(63,145),1
70 FOR X=63 TO 188
80 Y=INT(96+50*SIN(X/20))
90 PSET (X,Y),9
100 NEXT X
110 PRESET (90,170),0
120 PRINT #1,"SINUSCURVE"
130 GOTO 130
```

```
10 REM TANGENSCURVE
20 SCREEN 2
30 OPEN "GRP:" FOR OUTPUT AS #1
40 COLOR 1,15,15
50 LINE (63,96)-(188,96),1
60 LINE (63,36)-(63,155),1
70 FOR X=63 TO 188
80 Y=INT(96-50*TAN(X/20))
90 PSET (X,Y),9
100 NEXT X
110 PRESET (82,170),0
120 PRINT #1,"TANGENSCURVE"
130 GOTO 130
```

U moet er wel rekening mee houden dat de wiskundige formules voor de berekening van Y in regel 80 zijn aangepast om de juiste schermpositie voor Y te bepalen. In het programma hebben we het assenstelsel getekend met behulp van twee LINE-instructies. LINE heeft de volgende syntaxis:

LINE (beginpositie van X,Y)-(eindpositie van X,Y) ‹,kleur›

In bovenstaande programma's hebben we als kleur voor de assenstelsels zwart gekozen en voor de beide curves de kleur rood.

Schrijven in grafische schermen

In regel 110 bepalen we met PRESET de beginpositie van de tekst. Die tekst wordt dan met PRINT #1, op het scherm gePRINT. Voordat u in een grafisch scherm kunt schrijven, moet u eerst een kanaal naar dat scherm OPENen. In deze programma's gebeurde dat in regel 30:

OPEN "GRP:" FOR OUTPUT AS #1

In deze regel wordt kanaal #1 voor uitvoer naar het grafische scherm geOPENd. Een omslachtige methode, die ons meteen brengt bij de vraag waarom we in de spelletjes meestal SCREEN 0 of 1 gebruiken. Een grafisch scherm is uitstekend geschikt als u met sprites werkt, maar wilt u ook opnieuw gedefinieerde karakters gebruiken, dan is de bovenstaande methode dermate omslachtig dat de schermen 0 en 1 de voorkeur verdienen.

Hoofdstuk 20

Spelplezier

Om dit boek in stijl af te sluiten, volgen hier drie schitterende spelen. Het zijn allemaal variaties op de principes die we in dit boek hebben besproken. Deze programma's laten zien wat u met wat fantasie, een MSX en veel ijver kunt bereiken.

Safari

U bent jager op groot wild in de Afrikaanse jungle, maar helaas bent u verdwaald. U moet de weg naar het kamp terug zien te vinden zonder te worden opgegeten, aangevallen of verwond door één van de vele beesten. Sommige dieren blijven rustig grazen als u voorbijkomt, anderen hebben het duidelijk op u gemunt. Na RUN wordt het scherm blauw en volgt er een korte pauze waarin het spel wordt geïnitialiseerd. Het scherm wordt vervolgens rood en de titel verschijnt. Na een muzikaal intro worden de dieren die u op uw omzwervingen tegen kunt komen een voor een aan u voorgesteld. Let goed op gorilla's en neushoorns. Het scherm wordt nu lichtgroen en er verschijnen een paar groene struiken. Rechts op het veld is een schutting, die we gebruiken om het speelveld te verkleinen. In dit soort spelletjes, waarin voorwerpen naar u toe bewegen vanaf de bovenkant of onderkant van het scherm, is een schermbreedte van 32 posities (in SCREEN 1) vergeleken met de hoogte (24 posities) soms te groot. In dit geval heeft u teveel ruimte, dus hebben we ervoor gezorgd dat u niet het hele scherm, maar een gedeelte daarvan ter beschikking heeft. Als het scherm is opgebouwd, kunt u richting kamp gaan lopen. Succes.



```
10 REM *** SAFARI ***
20 GOSUB 1250
30 GOSUB 1690
40 GOSUB 1340
50 N=1:KL=1:Z=110:DO=0:SC=0:SPRITE ON
60 X=0:Y=195
70 VPOKE K+12,&B11001010
80 VPOKE K+15,&B11011010
90 FOR T=1 TO 23:PRINT TAB(INT(27*RND(1)+1))"
f":PRINT TAB(28) CHR$(30);"|":NEXT T
100 R=STICK(1)
110 V=STRIG(1)
120 IF R=1 THEN UP=0
130 IF R=5 THEN UP=100:Y=Y-10
140 IF R=7 THEN Z=Z-10:IF Z<20 THEN Z=20
150 IF R=3 THEN Z=Z+10:IF Z>220 THEN Z=220
160 IF OT<>100 THEN AP=INT(20*RND(1)+1):IF AP
=20 THEN OT=100
170 IF SH=100 THEN GOSUB 810
180 IF V<>0 THEN SH=100:A=Z:S=50:GOSUB 1030
190 IF OT=100 THEN GOSUB 610
```

```
200 PUT SPRITE 0,(Z,45),1,0
210 PUT SPRITE 1,(X,Y),KL,N
220 IF UP<>100 THEN 270
230 IF UP=100 THEN DO=DO+1
240 IF DO>1000 THEN GOTO 290
250 IF INT(3*RND(1)+1)=3 THEN PRINT TAB(INT(2
7*RND(1)+1));"f";
260 PRINT TAB(28);"|"
270 GOTO 100
280 REM *** HUT AAN HET EINDE ***
290 PRINT TAB(10)"ggg   ggg"
300 PRINT TAB(10)"g       g"
310 PRINT TAB(10)"g       g"
320 PRINT TAB(10)"g       g"
330 PRINT TAB(10)"g       g"
340 PRINT TAB(10)"g       g"
350 PRINT TAB(10)"ggggggggg"
360 SPRITE OFF
370 X=Z:FOR Y=45 TO 150:IF X<110 THEN X=X+4
380 IF X>110 THEN X=X-4
390 PUT SPRITE 0,(X,Y),1,0
400 NEXT Y
410 FOR TD=1 TO 1000:NEXT TD
420 REM *** SAFE ***
430 SCREEN 1
440 LOCATE 0,4
450 PRINT"GOED GEDAAN"
460 REM *** SCORE ***
470 LOCATE 0,8
480 PRINT"JE SCOORDE ";SC
490 FOR TD=1 TO 400:NEXT TD
500 LOCATE 0,10
510 PRINT"WIL JE NOG EENS OP SAFARI?"
520 A$=INKEY$
530 IF A$<>"N" AND A$<>"J" THEN GOTO 520
540 IF A$="J" THEN RUN
550 END
560 REM *** WELK DIER ***
570 IF DO<200 THEN AN=1:RETURN
580 IF DO<500 THEN AN=INT(RND(1)*2+1):RETURN
590 IF DO<700 THEN AN=INT(RND(1)*2+2):RETURN
600 AN=INT(RND(1)*2+3):RETURN
610 REM *** BEWEEG HET DIER ***
620 GOSUB 560
```

```
630 IF O=100 THEN 700
640 IF AN=1 THEN N=1:KL=4:GOTO 680
650 IF AN=2 THEN N=2:KL=1:GOTO 680
660 IF AN=3 THEN N=3:KL=15:GOTO 680
670 IF AN=4 THEN N=4:KL=6
680 O=100:IF INT(20*RND(1)+1)=20 THEN EX=100
690 X=INT(170*RND(1)+50):Y=192
700 IF CH=100 THEN 730
710 IF INT(5*RND(1)+1)=5 THEN CH=100
720 RETURN
730 Y=Y-(AN*5):X=X+INT(20*RND(1)+1)-10
740 IF Z>X THEN X=X+5*AN
750 IF Z<X THEN X=X-5*AN
760 IF X>212 THEN X=212
770 IF X<5 THEN X=5
780 IF Y<-14 THEN AP=0:AN=0:P=0:CH=0:O=0:OT=0
:EX=0
790 RETURN
800 REM *** KOGEL BEWEGEN ***
810 S=S+15:IF S>Y+40 THEN A=0:S=195:SH=0
820 PUT SPRITE 2,(A,S),1,5
830 RETURN
840 REM *** CONTROLE OP BOTSING ***
850 GOTO 100
860 SCREEN 1
870 COLOR 1,5,5
880 GOTO 460
890 A=0:S=195:IF SH=100 THEN GOSUB 1130
900 PUT SPRITE 1,(A,S),KL,N
910 SPRITE ON
920 X=INT(170*RND(1)+50):Y=192
930 GOTO 100
940 REM *** DODENMUZIEK ***
950 RESTORE 2270
960 SOUND 7,62
970 FOR M=1 TO 15
980 READ E
990 READ D
1000 PLAY"L=D;N=E;"
1010 NEXT M
1020 RETURN
1030 REM *** SCHOT GELUID ***
1040 SOUND 0,200
1050 SOUND 1,5
```

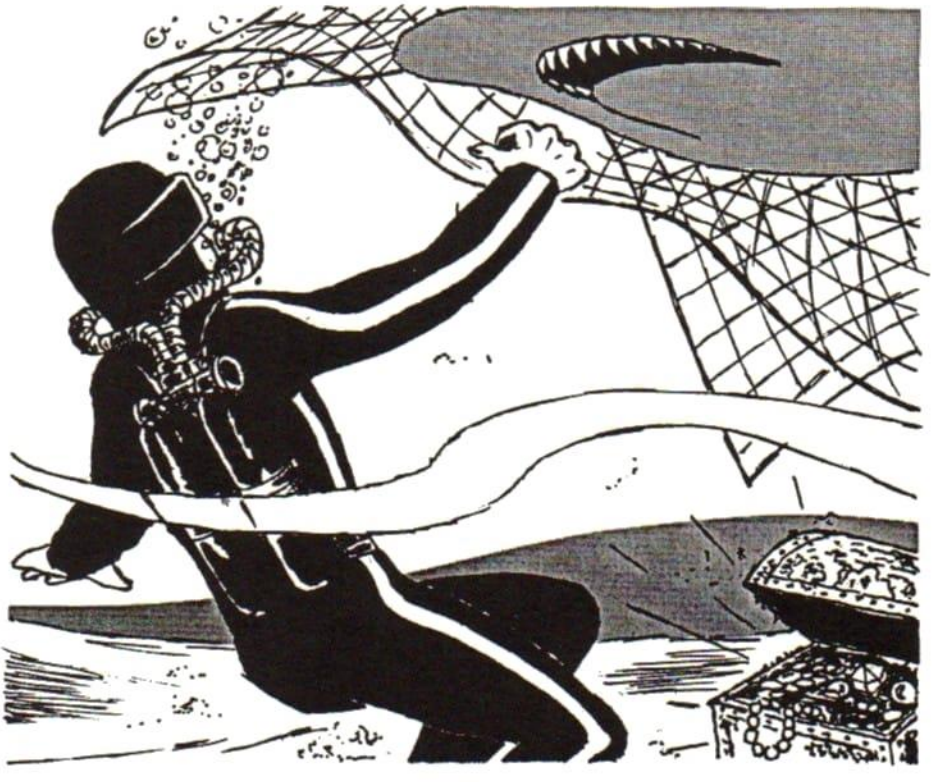
```
1060 SOUND 6,20
1070 SOUND 7,55
1080 SOUND 8,16
1090 SOUND 11,5
1100 SOUND 12,30
1110 SOUND 13,0
1120 RETURN
1130 REM *** GELUID VAN EEN GERAAKT DIER ***
1140 FOR H=255 TO 5 STEP -50
1150 SOUND 0,H
1160 SOUND 1,10
1170 SOUND 7,62
1180 SOUND 8,13
1190 NEXT H
1200 SOUND 7,63
1210 RETURN
1220 SOUND 1,0
1230 RETURN
1240 END
1250 REM *** TITELPAGINA ***
1260 CLS
1270 COLOR 1,10,6
1280 SCREEN 3
1290 OPEN "GRP:" FOR OUTPUT AS #1
1300 PRESET (45,80),10
1310 PRINT#1,"SAFARI"
1320 FOR TD=1 TO 2000:NEXT TD
1330 RETURN
1340 REM *** EVEN VOORSTELLEN ***
1350 LOCATE 3,12
1360 FOR I=1 TO 5
1370 ON I GOSUB 1540,1570,1600,1630,1660
1380 FOR Y=195 TO -15 STEP-1
1390 X=128
1400 PUT SPRITE 0,(X,Y),KL,I-1
1410 FOR TD=1 TO 2:NEXT TD
1420 NEXT Y
1430 LOCATE 3,12
1440 NEXT I
1450 RETURN
1460 REM *** BOTSING ***
1470 SPRITE OFF
1480 IF A<X+15 AND A>X-15 AND S<Y+15 AND S>Y-
15 THEN SC=SC+(AN*50)+EX ELSE GOTO 1510
```

```
1490 X=0:Y=192:OT=0
1500 GOTO 890
1510 IF X<Z+15 AND X>Z-15 AND Y<55 AND Y>35 T
HEN GOSUB 940:GOTO 860
1520 SPRITE ON
1530 RETURN
1540 PRINT "JIJ!!"
1550 KL=1
1560 RETURN
1570 PRINT "NIJLPAARD"
1580 KL=4
1590 RETURN
1600 PRINT "GORILLA  "
1610 KL=1
1620 RETURN
1630 PRINT "NEUSHOORN"
1640 KL=15
1650 RETURN
1660 PRINT "LEEUW    "
1670 KL=6
1680 RETURN
1690 REM *** INITIALISATIE ***
1700 SCREEN 1,2
1710 WIDTH 32
1720 KEY OFF
1730 CLS
1740 K=8192
1750 ON SPRITE GOSUB 1460
1760 REM *** SPRITE DEFINIEREN ***
1770 RESTORE 1930
1780 FOR I=0 TO 5
1790 FOR B=1 TO 32
1800 READ S
1810 S$=S$+CHR$(S)
1820 NEXT B
1830 SPRITE$(I)=S$:S$=""
1840 NEXT I
1850 REM *** KARAKTERS 102 EN 103 HERDEFINIER
EN ***
1860 FOR N=102 TO 103
1870 FOR M=0 TO 7
1880 READ S
1890 VPOKE M+N*8,S
1900 NEXT M
```

```
1910 NEXT N
1920 RETURN
1930 REM *** DATA VOOR SPRITES ***
1940 REM *** JIJ ***
1950 DATA 15,127,127,127,51,27,15,7
1960 DATA 3,3,3,3,3,3,3,3
1970 DATA 224,248,248,248,248,216,152,24
1980 DATA 48,96,192,128,0,0,0,0
1990 REM *** NIJLPAARD ***
2000 DATA 3,7,7,3,7,15,15,15
2010 DATA 7,15,31,31,47,55,41,7
2020 DATA 192,224,224,192,224,240,240,240
2030 DATA 224,240,248,248,240,224,128,0
2040 REM *** GORILLA ***
2050 DATA 0,0,0,0,0,0,112,113
2060 DATA 115,119,127,63,31,7,0,0
2070 DATA 0,0,0,0,14,14,14,142
2080 DATA 206,238,252,248,240,224,0,0
2090 REM *** NEUSHOORN ***
2100 DATA 1,3,3,7,3,7,15,31
2110 DATA 31,31,31,31,31,15,7,3
2120 DATA 0,128,128,192,128,192,224,240
2130 DATA 240,240,240,240,224,192,152,240
2140 REM *** LEEUW ***
2150 DATA 1,3,3,7,3,7,11,22
2160 DATA 13,27,6,27,19,15,1,3
2170 DATA 0,128,128,192,128,64,160,144
2180 DATA 160,176,208,176,132,202,178,192
2190 REM *** DATA KOGEL ***
2200 DATA 0,0,0,0,0,0,0,1
2210 DATA 1,0,0,0,0,0,0,0
2220 DATA 0,0,0,0,0,0,0,128
2230 DATA 128,0,0,0,0,0,0,0
2240 REM *** DATA VOOR NIEUWE KARAKTERS ***
2250 DATA 16,56,84,254,84,16,56,0
2260 DATA 255,255,255,255,255,255,255,255
2270 REM *** DATA VOOR DODENMUZIEK ***
2280 DATA 31,4,32,5,32,4,33,3,32,5
2290 DATA 39,4,35,5,35,4,38,4,35,3
2300 DATA 21,3,31,4,34,5,25,4,29,2
```

JAWS

In dit spel hebt u de gelegenheid in de diepten van de oceaan te zoeken naar verborgen schatten. Helaas wordt uw leven bedreigd door de schrik van de oceaan, een grote witte reuzenhaai. Als u geluk hebt kunt u een schatkist vinden in een van de vele grotten. Soms ligt de kist verborgen en soms kunt hem zien. Het spel heeft twee schermen. Als u begint, zwemt u midden op een grote watervlakte. In de linker bovenhoek van het scherm dobbert uw bootje. In de buurt van het bootje is een klein puntje te zien: dat bent u. U moet van de boot wegzwemmen in de richting van de grotten rechts op het scherm. De cirkels zijn de grotten. Verder is er nog een joekel van een haai. Op het titelscherm is dat de asterisk. U moet hem zien te ontlopen en proberen de grotten te bereiken. Eenmaal in de grot aangeland, verschijnt het tweede scherm. In de grot moet u de schat zoeken. Als u niet blijft zwemmen, drijft u langzaam naar het zeeoppervlak. Hebt u de schat gevonden dan ziet u u dat in de linker bovenhoek van het scherm. Verlaat de grot en neem de schat mee naar de boot. Als de haai u op de terugweg te pakken krijgt, bent u de schat kwijt. Met een druk op de knop kunt u het net loslaten.



```
10 REM *** JAWS ***
20 BB=4:NN=4:DV=4:X=128:Y=30:LF=3
30 SPRITE ON
40 GOSUB 1880
50 IF ED=100 THEN FOR T=1 TO 10:CL(T)=0:NEXT
T:QR=0
60 GOSUB 770:GOSUB 320
70 R=STICK(1)
80 V=STRIG(1)
90 IF R=1 THEN Y=Y-6:DV=4:IF Y<30 THEN 50
100 IF R=5 THEN Y=Y+6:DV=4:IF Y>180 THEN Y=18
0
110 IF R=7 THEN X=X-6:DV=2:IF X<20 AND CR=0 T
HEN X=20
120 IF R=3 THEN X=X+6:DV=3:IF X>250 THEN CR=1
00
130 IF R=0 THEN Y=Y-2:DV=4:IF Y<30 THEN Y=30
140 IF APP=0 THEN T=INT(5*RND(1)+1):IF T=5 TH
EN 510
150 IF APP=100 THEN 550
160 IF APP<>100 THEN I=0:O=195
170 IF CR=0 THEN 200
180 IF CR=100 THEN X=8:CR=200
190 IF X<8 AND CR=200 THEN X=250:CR=0
200 IF SH<>0 THEN 220
210 IF V<>0 THEN SH=100:X3=X:Y3=Y:GOSUB 1470
220 IF SH=0 THEN GOTO 280
230 IF DV=4 THEN Y3=Y3-10:IF Y3<30 THEN SH=0
240 IF DV=3 THEN X3=X3-10:IF X3<20 THEN SH=0
250 IF DV=4 THEN X3=X3+10:IF X3>250 THEN 500
260 IF SH=0 THEN 420
270 PUT SPRITE 2,(X3,Y3),9,6
280 PUT SPRITE 1,(X,Y),1,DV
290 PUT SPRITE 0,(I,O),15,S
300 SPRITE ON
310 GOTO 70
320 REM *** MAAK SCHERM KLAAR ***
330 CLS
340 COLOR 1,5,5
350 VPOKE 8192+12,&B10110100
360 VPOKE 8192+13,&B10111100
370 FOR T=1 TO (4*29):PRINT"a";:NEXT T
380 GOSUB 670
390 GOSUB 1140
```

```
400 RETURN
410 REM *** NET ACHTER OBSTAKEL ***
420 CB=100
430 IF DV=4 THEN Y3=Y3-10:IF Y<50 THEN SH=0
440 IF DV=2 THEN X3=X3-10:IF X3<10 THEN 490
450 IF DV=3 THEN X3=X3+10:IF X3>80 THEN SH=0
460 IF SH<>0 THEN 270
470 CB=0
480 GOTO 270
490 X3=250:CB=0:DV=2:GOTO 260
500 X3=10:CB=100:GOTO 260
510 REM *** DE HAAI VERSCHIJNT ***
520 GOSUB 1660:T=INT(2*RND(1)+1):IF T=1 THEN
S=0:I=15:O=INT(115*RND(1)+50)
530 IF T=2 THEN S=1:I=250:O=INT(115*RND(1)+50
)
540 APP=100
550 REM *** HAAI BEWEGEN ***
560 IF AB=100 THEN S=1:GOTO 610
570 IF Y>O THEN O=O+3
580 IF Y<O THEN O=O-3
590 IF X>I THEN S=0
600 IF X<I THEN S=1
610 IF S=1 THEN I=I-3
620 IF S=0 THEN I=I+3
630 IF I>250 THEN AB=100:I=6:GOTO 650
640 IF I<6 AND AB=100 THEN 660
650 GOTO 160
660 I=250:AB=0:GOTO 160
670 REM *** ZEEBODEM ***
680 P=INT(5*RND(1)+1):ON P GOTO 690,710,730,7
50
690 LOCATE 0,21:PRINT"h      h  h h h   h  h
 h h"
700 PRINT"hhh      hhhhhhh  hhhhhhhhh":RETURN
710 LOCATE 0,21:PRINT"h   hh    h   h  h  hh
 h  "
720 PRINT"hhhh     hh  hhh   hhh hhh ":RETURN
730 LOCATE 0,21:PRINT"h  h  h  h  h  h  h  h
 h  "
740 PRINT" hh hh hh hh hh hh hh hh hh":RETURN
750 LOCATE 0,21:PRINT"h h h  hhh hh  h h h  h
h  h"
760 PRINT"hhhh h hhh hhhhhh hhhh hhh ":RETURN
```

```
770 REM *** AAN DE OPPERVLAKTE ***
780 CLS
790 FOR Z=0 TO 6:PUT SPRITE Z,(0,195),0,Z:NEX
T Z
800 COLOR 15,6,6
810 LOCATE 1,1:PRINT CHR$(208);CHR$(219);CHR$
(207)
820 VPOKE 6144+20+5*32,9
830 VPOKE 6144+22+7*32,9
840 VPOKE 6144+19+12*32,9
850 VPOKE 6144+19+16*32,9
860 VPOKE 6144+30+19*32,9
870 FOR T=1 TO 5:VPOKE CL(T),32:NEXT T:IF QR=
5 THEN ED=100
880 B=BB:N=NN:B2=20*RND(1):N2=20*RND(1)
890 VPOKE 6144+B+32*N,32
900 R=STICK(1)
910 IF R=1 THEN N=N-1:IF N<1 THEN N=1
920 IF R=5 THEN N=N+1:IF N>23 THEN N=23
930 IF R=7 THEN B=B-1:IF B<1 THEN B=1
940 IF R=3 THEN B=B+1:IF B>30 THEN B=30
950 FI=VPEEK(6144+B+32*N):IF FI=9 THEN 1080
960 IF FI=219 AND PO=100 THEN PO=0:GOTO 1100
970 VPOKE 6144+B+32*N,46
980 IF DAP=100 THEN 890
990 VPOKE 6144+INT(B2)+32*INT(N2),32
1000 IF B>INT(B2) THEN B2=B2+.5:IF B2>32 THEN
 B2=32
1010 IF B<INT(B2) THEN B2=B2-.5:IF B2>0 THEN
B2=0
1020 IF N>INT(N2) THEN N2=N2+.5:IF N2>23 THEN
 N2=23
1030 IF N<INT(N2) THEN N2=N2-.5:IF N2<0 THEN
N2=0
1040 AR=VPEEK(6144+INT(B2)+32*INT(N2)):IF AR=
9 THEN DAP=100:SC=SC+50:GOTO 890
1050 IF AR=46 THEN 1280
1060 VPOKE 6144+INT(B2)+32*INT(N2),42
1070 GOTO 890
1080 CL(QR)=6144+B+32*N:CR=0
1090 DV=3:X=INT(200*RND(1)+50):Y=30:S=0:APP=0
:DAP=0:BB=B+1:NN=N:RETURN
1100 T=INT(100*RND(1)+50)*R
1110 PRINT"GOED GEDAAN"
```

```
1120 PRINT"DE SCHAT IS WAARD ";T:SC=SC+T:FOR
TD=1 TO 1000:NEXT TD
1130 GOSUB 1850:BB=4:NN=2:DAP=0:GOTO 50
1140 REM *** ZET SCHAT OP DE BODEM ***
1150 C=INT(200*RND(1)+50):KL=INT(16*RND(1))
1160 PUT SPRITE 4,(C,175),KL,5
1170 RETURN
1180 REM *** CONTROLE OP BOTSINGEN ***
1190 SPRITE OFF
1200 IF X<I+20 AND X>I-20 AND Y<O+20 AND Y>O-
20 THEN GOTO 1280
1210 IF X3<I+20 AND X3>I-20 AND Y3<O+20 AND Y
3>O-20 AND SH>0 THEN APP=0:S=0:GOTO 1260
1220 IF X<C+15 AND X>C-15 AND Y>185 THEN GOTO
 1240
1230 GOTO 70
1240 IF PO<>100 THEN LOCATE 0,0:PRINT"SCHAT":
GOSUB 1750
1250 PO=100:GOTO 1230
1260 GOSUB 1570
1270 FOR T=O TO 221 STEP RND(1)+.2:PUT SPRITE
 7,(T,O),1,7:PUT SPRITE 0,(T,O),1,S:NEXT T:RE
TURN
1280 REM *** EEN LEVEN VERLOREN ***
1290 SOUND 7,62:PLAY "T255BAGFEDC":FOR S=1 TO
 5:FOR T=0 TO 15:COLOR ,T,T:NEXT T:NEXT S:COL
OR 15,4,4:PO=0
1300 LF=LF-1:IF LF<1 THEN GOTO 1320
1310 CL(QR)=0:QR=QR+1:GOTO 50
1320 REM *** EINDE TIJD ***
1330 SCREEN 1
1340 COLOR 1,11,11
1350 LOCATE 2,10
1360 PRINT"JE BENT DE HAAI TE VAAK"
1370 LOCATE 2,11
1380 PRINT"TEGENGEKOMEN"
1390 LOCATE 2,12
1400 PRINT"JE SCORE IS:";SC
1410 LOCATE 2,13
1420 PRINT"WIL JE NOG EENS?"
1430 A$=INKEY$
1440 IF A$="J" THEN RUN
1450 IF A$="N" THEN END
1460 GOTO 1430
```

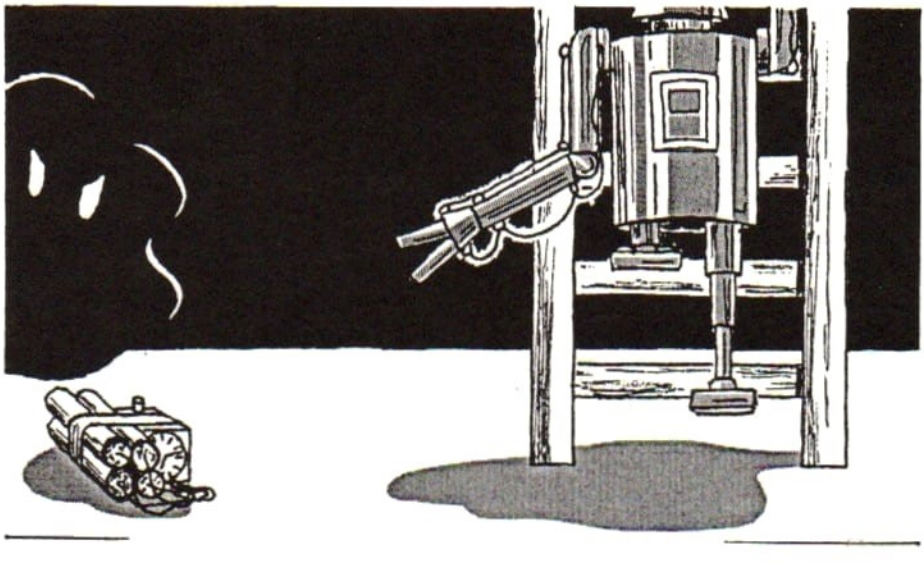
```
1470 REM *** GELUID ***
1480 REM *** SCHOT ***
1490 SOUND 0,12
1500 SOUND 1,3
1510 SOUND 6,20
1520 SOUND 7,55
1530 SOUND 8,16
1540 SOUND 13,0
1550 FOR TD=1 TO 100:NEXT TD
1560 RETURN
1570 REM *** HAAI GERAAKT ***
1580 SOUND 0,154
1590 SOUND 1,0
1600 SOUND 6,20
1610 SOUND 7,55
1620 SOUND 8,8
1630 SOUND 13,0
1640 FOR TD=1 TO 100:NEXT TD
1650 RETURN
1660 REM *** HAAI VERSCHIJNT ***
1670 SOUND 0,111
1680 SOUND 2,3
1690 SOUND 6,10
1700 SOUND 7,55
1710 SOUND 8,16
1720 SOUND 13,0
1730 FOR TD=1 TO 100:NEXT TD
1740 RETURN
1750 REM *** DE SCHAT ***
1760 SOUND 0,12
1770 SOUND 2,3
1780 SOUND 6,20
1790 SOUND 7,55
1800 SOUND 8,16
1810 SOUND 13,0
1820 FOR TD=1 TO 100:NEXT TD
1830 REM BEETJE GELUID
1840 RETURN
1850 REM *** DE SCHAT NAAR DE BOOT ***
1860 SOUND 7,62:PLAY "CEGCEG."
1870 RETURN
1880 REM *** INITIALISATIE ***
1890 KEY OFF
1900 CLS
```

```
1910 Z=RND(-TIME)
1920 SCREEN 1,2
1930 ON SPRITE GOSUB 1180
1940 REM *** SPRITE DEFINIERING ***
1950 FOR I=0 TO 6
1960 FOR Q=1 TO 32
1970 READ A
1980 S$=S$+CHR$(A)
1990 NEXT Q
2000 SPRITE$(I)=S$:S$=""
2010 NEXT I
2020 REM *** HERDEFINIEREN KARAKTER ***
2030 FOR N=97 TO 104 STEP 7
2040 FOR M=0 TO 7
2050 READ A
2060 VPOKE N*8+M,A
2070 NEXT M
2080 NEXT N
2090 RETURN
2100 REM *** DATA SPRITES ***
2110 REM *** HAAI > ***
2120 DATA 0,0,0,1,1,129,79,127
2130 DATA 127,71,128,0,0,0,0,0
2140 DATA 0,0,0,0,128,240,252,255
2150 DATA 252,240,0,0,0,0,0,0
2160 REM *** HAAI < ***
2170 DATA 0,0,0,0,1,15,63,255
2180 DATA 63,15,0,0,0,0,0,0
2190 DATA 0,0,0,128,128,129,242,254
2200 DATA 254,226,1,0,0,0,0,0
2210 REM *** DUIKER < ***
2220 DATA 0,0,0,0,7,111,255,255
2230 DATA 111,7,0,0,0,0,0,0
2240 DATA 0,0,0,0,192,224,254,255
2250 DATA 227,193,1,0,0,0,0,0
2260 REM *** DUIKER > ***
2270 DATA 0,0,0,0,3,7,255,255
2280 DATA 199,131,128,0,0,0,0,0
2290 DATA 0,0,0,0,224,246,255,255
2300 DATA 246,224,0,0,0,0,0,0
2310 REM *** DUIKER ^ ***
2320 DATA 1,3,3,1,7,15,23,39
2330 DATA 7,7,3,3,3,2,6,6
2340 DATA 128,192,192,128,224,240,232,228
```

```
2350 DATA 224,224,192,192,192,64,96,96
2360 REM *** SCHAT ***
2370 DATA 0,0,0,0,0,0,0,0
2380 DATA 0,0,0,0,0,0,0,0
2390 DATA 0,0,0,0,0,0,0,0
2400 DATA 0,60,126,255,255,255,255,0
2410 REM *** NET ***
2420 DATA 13,21,36,66,217,34,148,40
2430 DATA 148,34,241,138,100,40,22,0
2440 DATA 176,104,68,218,41,205,80,45
2450 DATA 80,137,13,178,67,34,244,0
2460 REM *** DATA VAN KARAKTERS ***
2470 DATA 255,255,255,255,255,255,255,255
2480 DATA 255,255,255,255,255,255,255,255
```

ROBOT

ROBOT, een spel waarvoor behendigheid en durf is vereist, daagt u uit om een tijdbom onschadelijk te maken voor deze ontploft. We hebben gebruik gemaakt van zelfbedachte tekens en een paar fantasievolle afbeeldingen op het scherm. Een prima spel is het resultaat. De bedoeling van het spel is om een kleine robot over het scherm te bewegen. Daarbij klimt de robot ladders op en af om onderdelen te verzamelen die nodig zijn om de bom onschadelijk te maken. Als u alle onderdelen bij elkaar heeft, moet u op de bom afsnellen en hem onschadelijk maken. Om het u nog moeilijker te maken hebben we er een vormeloze amoebe bijgedaan die over het scherm zweeft en u probeert tegen te houden. U beweegt de robot met de joystick. Door een druk op de knop wordt u onzichtbaar. Als u dit doet, verliest de amoebe u uit het oog. Maar pas op! Als u onzichtbaar bent, kunt u gemakkelijk van een ladder of een platform aflopen en naar beneden vallen.



Er zijn drie niveaus. Niveau 1 is het gemakkelijkst en niveau 3 het moeilijkst. Bij een oplopende moeilijkheidsgraad moet u meer onderdelen ophalen. Ook hebt u meer te stellen met de afbeeldingen op het scherm. Robot is een pittig spel. Waarschijnlijk heeft u al de grootste moeite met niveau 1. Onze eigen experts op het gebied van computerspelen hebben op niveau 3 de opdracht nog niet kunnen voltooien.

```
10 REM *** ROBOT ***
20 SCREEN 1
30 KEY OFF
40 K=8192
50 CLS:LOCATE 5,12
60 PRINT "WELK NIVEAU (1-3)";
70 A$=INKEY$:IF A$="" THEN 70
80 IF ASC(A$)>51 OR ASC(A$)<49 THEN 70
90 LV=VAL(A$):PRINT LV
100 COLOR 1,11,11
110 CLS
120 LOCATE 9,12
130 PRINT "EVEN GEDULD"
140 LF=3
150 GOSUB 390
160 JV=STICK(1):VK=STRIG(1)
170 IF CG=100 AND JV=0 THEN FOR TV=1 TO 100:N
EXT TV:GOTO 310
180 IF JV=0 THEN 200
190 ST=JV
200 IF ST=1 THEN Y=Y-1:CHAR=145
210 IF ST=5 THEN Y=Y+1:CHAR=145
220 IF ST=7 THEN X=X-1:CHAR=161
230 IF ST=3 THEN X=X+1:CHAR=153
240 CR=VPEEK(6144+X+32*Y)
250 IF CR=32 AND ST<>1 AND ST<>5 THEN GOSUB 2
070:GOTO 1350
260 IF CR=32 OR CR=61 THEN X=X2:Y=Y2:CG=100:G
OTO 300
270 CG=0
280 IF CR=105 THEN CNO=CNO+1:CR=121:SC=SC+100
:GOSUB 2030:IF CNO=1 THEN TIME=0
290 IF CR=97 THEN 1450
300 IF VK=-1 THEN CHAR=CR:SC=SC-.25
310 GOSUB 1620
```

```
320 VPOKE 6144+X+32*Y,CHAR:IF CG=100 THEN 340
330 VPOKE 6144+X2+32*Y2,CR
340 IF CNO=0 THEN 370
350 MIN%=TIME/3600:SEC%=TIME/60-(MIN%*60)
360 MM=SEC%:TM=INT(MM/2):TM=25-TM:VPOKE 6144+
TM,32:IF TM=0 THEN RES=1:GOTO 1300
370 LOCATE 23,20:PRINT INT(SC)
380 X2=X:Y2=Y:GOTO 160
390 REM TEKENS
400 FOR C=97 TO 161 STEP 8
410 FOR M=0 TO 7
420 READ A
430 VPOKE C*8+M,A
440 NEXT M
450 NEXT C
460 DATA 4,8,8,60,126,126,126,60:REM BOM
470 DATA 192,192,32,16,8,4,2,1:REM MINEUR
480 DATA 204,51,204,51,204,51,204,51:REM STEL
LAGE
490 DATA 0,0,0,0,255,255,102,102:REM STELLAGE
500 DATA 195,255,255,195,195,255,255,195:REM
TRAP
510 DATA 129,189,102,231,255,126,189,129:REM
BOBBEL
520 DATA 24,24,60,255,189,189,126,126:REM ROB
OT OMHOOG
530 DATA 48,48,120,94,91,72,252,252:REM ROBOT
 NAAR RECHTS
540 DATA 12,12,30,250,26,18,63,63:REM ROBOT N
AAR LINKS
550 GOTO 1110
560 ON LV GOTO 570,740,940
570 REM SCHERM 1
580 COLOR 1,3,3
590 VPOKE K+13,67
600 VPOKE K+14,211
610 VPOKE K+15,211
620 VPOKE K+16,179
630 VPOKE K+17,131
640 FOR T=1 TO 31:VPOKE 6144+T+32*18,121:NEXT
650 FOR P=1 TO 10:PRINT:NEXT:PRINT TAB(4)"qqq
qqqqqq        qqqqqqq"
660 FOR T=11 TO 17:VPOKE 6144+12+32*T,129:NEX
T
```

```
670 FOR T=11 TO 17:VPOKE 6144+24+32*T,129:NEX
T
680 LOCATE 25,7:PRINT "qqq"
690 FOR T=4 TO 9:VPOKE 6144+14+32*T,129:NEXT
700 FOR T=8 TO 9:VPOKE 6144+28+32*T,129:NEXT
710 LOCATE 5,1:PRINT "qqqq":LOCATE 8,2:PRINT
"qq":LOCATE 9,3:PRINT "qq":LOCATE 10,4:PRINT
"qqq":VPOKE 6144+38,97
720 VPOKE 6144+607,105
730 VPOKE 6144+250,105:GOTO 1270
740 REM *** SCHERM 2 ***
750 COLOR 15,1,1
760 VPOKE K+13,177
770 VPOKE K+14,145
780 VPOKE K+15,209
790 VPOKE K+16,81
800 VPOKE K+17,113
810 FOR T=1 TO 31:VPOKE 6144+T+32*18,121:NEXT
820 LOCATE 20,2:PRINT"qqqqq"
830 LOCATE 20,6:PRINT"qqqqqqqq"
840 LOCATE 8,11:PRINT"qqqqqqqqqqq"
850 LOCATE 9,18:PRINT"  qqqq   "
860 FOR T=3 TO 5:VPOKE 6144+22+T*32,129:NEXT
870 FOR T=7 TO 17:VPOKE 6144+29+T*32,129:NEXT
880 FOR T=12 TO 17:VPOKE 6144+10+T*32,129:NEX
T
890 FOR T=12 TO 17:VPOKE 6144+15+T*32,129:NEX
T
900 FOR T=12 TO 17:VPOKE 6144+20+T*32,129:NEX
T
910 VPOKE 6144+1+32*18,105:VPOKE 6144+8+32*18
,105
920 VPOKE 6144+23+32*18,105:VPOKE 6144+24+6*3
2,105
930 VPOKE 6144+26+2*32,97:GOTO 1270
940 REM *** SCHERM 3 ***
950 COLOR 1,4,4
960 VPOKE K+13,228
970 VPOKE K+14,116
980 VPOKE K+15,196
990 VPOKE K+16,244
1000 VPOKE K+17,164
1010 FOR T=1 TO 31:VPOKE 6144+T+32*18,121:NEX
T
```

```
1020 LOCATE 1,3:PRINT"qqqqqqqqqqqqqqqqqqqqqqq
qqqq"
1030 LOCATE 1,7:PRINT"qqqqqqqqqqqqi  iqqqqqqq
qqqq"
1040 LOCATE 1,11:PRINT"qqqqqqqqqi        iqqq
qqqqq"
1050 LOCATE 1,15:PRINT"qqqqqi     qqqqq
iqqqq"
1060 FOR T=2 TO 17:VPOKE 6144+2+T*32,129:NEXT
1070 FOR T=2 TO 17:VPOKE 6144+30+T*32,129:NEX
T
1080 FOR T=16 TO 17:VPOKE 6144+14+32*T,129:VP
OKE 6144+18+32*T,129:NEXT
1090 VPOKE 6144+16+14*32,97
1100 GOTO 1270
1110 REM *** START ***
1120 CLS:X(1)=14:X(2)=15:X(3)=16:X(4)=17:REM
SCHOON SCHERM
1130 FOR Y=0 TO 22
1140 FOR T=1 TO 4:X(T)=X(T)+1:IF X(T)>17 THEN
 X(T)=14
1150 NEXT
1160 CLS:REM SCHOON SCHERM
1170 VPOKE 6144+X(1)+32*Y,121:VPOKE 6144+X(2)
+32*Y,121:VPOKE 6144+X(3)+32*Y,129
1180 VPOKE 6144+X(4)+32*Y,121
1190 VPOKE 6144+X(3)+32*(Y-1),145
1200 FOR T=1 TO 4:IF T/3=INT(T/3) THEN GOTO 1
220
1210 VPOKE K+1,1
1220 NEXT:NEXT
1230 Y=Y-1
1240 VPOKE 6144+(X(1)-4)+32*Y,82:VPOKE 6144+X
(2)+32*Y,79:VPOKE 6144+X(3)+32*Y,66
1250 VPOKE 6144+X(4)+32*Y,79:VPOKE 6144+(X(4)
+1)+32*Y,84
1260 FOR TD=1 TO 2000:NEXT TD:GOTO 560
1270 X=16:Y=18:X2=X:Y2=Y:CHAR=35:ST=1:CNO=0:D
ISS=0:DR=0:Q=0:W=0:APP=0:PC=32
1280 LOCATE 0,20:PRINT "ROBOTS:";LF;"
SCORE:"
1290 LOCATE 0,0:PRINT "==================
==":TIME=0:RETURN
1300 REM EXPLOSIE
```

```
1310 GOSUB 2130:FOR T=1 TO 50:FOR C=15 TO 16:
VPOKE 6144+INT(RND(1)*1000),32
1320 VPOKE 6144+INT(1000*RND(1)),32:VPOKE 614
4+INT(1000*RND(1)),32
1330 NEXT:NEXT
1340 FOR TV=1 TO 1500:NEXT TV:CLS:GOTO 1850
1350 REM *** VAL VAN DE RAND ***
1360 VPOKE 6144+X2+32*Y2,32:LF=LF-1:IF LF<1 T
HEN RES=3:GOTO 1850
1370 FOR T=Y TO 22:VPOKE 6144+X+32*(T-1),32
1380 VPOKE 6144+X+32*T,145:FOR N=1 TO 100:NEX
T:NEXT
1390 CLS:IF X=15 THEN 1430
1400 VPOKE 6144+X+21*32,32:IF X<15 THEN X=X+1
:CHAR=153:IF X=15 THEN 1430
1410 IF X>15 THEN X=X-1:CHAR=161:IF X=15 THEN
 1430
1420 VPOKE 6144+X+21*32,CHAR:GOTO 1400
1430 VPOKE 6144+687,145
1440 LOCATE 11,22:PRINT "ROBOT":LOCATE 0,0:GO
SUB 560:GOTO 160
1450 REM ONTMANTELEN
1460 IF CNO<>(LV*2) THEN RES=2:GOTO 1300
1470 GOSUB 2100:REM MUZIEK:GOSUB 875
1480 SC=SC+(1000*LV)+(TM*10)
1490 FOR TD=1 TO 1000:NEXT TD:CLS
1500 VPOKE 6144+X+Y,32
1510 IF X>15 THEN X=X-1:CHAR=161
1520 IF X<15 THEN X=X+1:CHAR=153
1530 VPOKE 6144+X+32*Y,CHAR
1540 IF X=15 THEN FOR T=1 TO 500:NEXT:GOTO 15
60
1550 GOTO 1500
1560 VPOKE 6144+X+32*Y,32:Y=Y+1:IF Y=18 THEN
VPOKE 6144+100,145:GOTO 1580
1570 VPOKE 6144+X+32*Y,145:GOTO 1560
1580 CLS:LOCATE 0,22:PRINT TAB(14)"ROBOT":LOC
ATE 0,0
1590 VPOKE 6144+687,145
1600 LV=LV+1:IF LV>3 THEN LV=1
1610 GOSUB 560:GOTO 160
1620 REM DE TOORNIGE TOR
1630 IF APP=100 THEN 1690
1640 T=INT(5*RND(1)+1):IF T<(5-LV+1) THEN FOR
```

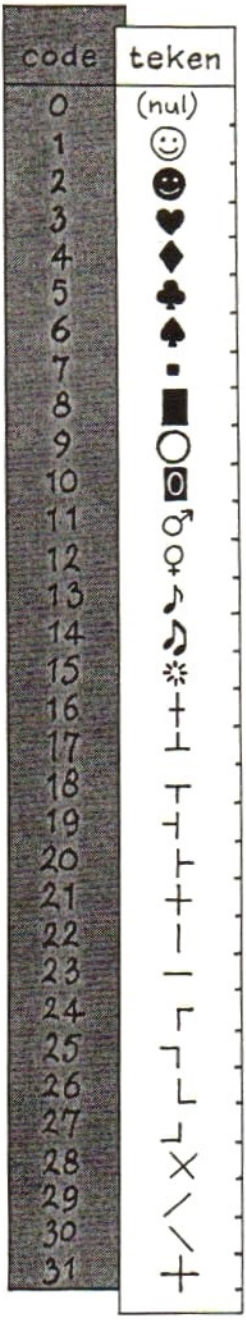
```
 T=1 TO 10:NEXT:RETURN
1650 APP=100:DR=INT(3*RND(1)+1)
1660 IF DR=1 THEN Q=INT(30*RND(1)+1):W=0
1670 IF DR=2 THEN Q=30:W=INT(18*RND(1)+1)
1680 IF DR=3 THEN Q=0:W=INT(18*RND(1)+1)
1690 VPOKE 6144+Q+32*W,PC
1700 IF VK<>0 THEN 1810
1710 T=INT(3*RND(1)+1):IF T<3 THEN 1810
1720 IF Q<X THEN Q=Q+1:IF Q>30 THEN APP=0
1730 IF Q>X THEN Q=Q-1:IF Q<0 THEN APP=0
1740 IF W<Y THEN W=W+1:IF W>18 THEN APP=0
1750 IF W>Y THEN W=W-1:IF W<0 THEN APP=0
1760 PC=VPEEK(6144+Q+32*W)
1770 IF PC=145 OR PC=153 OR PC=161 THEN GOSUB
 2070:GOTO 1350
1780 VPOKE 6144+Q+32*W,137
1790 IF APP=0 THEN VPOKE 6144+Q+(32*W),32:SC=
SC+10:PC=32
1800 RETURN
1810 IF DR=1 THEN W=W+1:IF W>18 THEN APP=0
1820 IF DR=2 THEN Q=Q-1:IF Q<0 THEN APP=0
1830 IF DR=3 THEN Q=Q+1:IF Q>30 THEN APP=0
1840 GOTO 1760
1850 REM *** EINDE SEQUENTIE
1860 CLS:SC=INT(SC)
1870 ON RES GOTO 1880,1890,1900
1880 LOCATE 0,5:PRINT "JE TIJD IS OP!!":GOTO
1910
1890 LOCATE 0,5:PRINT "JE HEBT NIET ALLE MINE
URS":GOTO 1910
1900 LOCATE 0,5:PRINT "JE BENT EEN KEER TE VA
AK":PRINT "GEVALLEN"
1910 LOCATE 0,10:PRINT "JE SCOORDE";SC;"PUNTE
N"
1920 LOCATE 0,20:PRINT "NOG EEN KEER?";
1930 A$=INKEY$
1940 IF A$<>"J" THEN GOTO 1960
1950 GOTO 1980
1960 IF A$<>"N" THEN GOTO 1930
1970 END
1980 CLS:LOCATE 8,5:PRINT "WELK NIVEAU?"
1990 A$=INKEY$:IF A$<>"1" AND A$<>"2" AND A$<
>"3" THEN 1990
2000 LV=VAL(A$):CLS:LF=3:SC=0
```

```
2010 GOSUB 560:GOTO 160
2020 REM SOUND
2030 REM PAK DE MINEUR
2040 PLAY "V15 L64 C+D#E#F#G#A#B"
2050 PLAY "V15 L64 B+A#G#F#E#D#C"
2060 RETURN
2070 REM VAL ERAF
2080 SOUND 0,127:SOUND 7,56:SOUND 8,16:SOUND
13,25
2090 RETURN
2100 REM ONTMANTELEN
2110 PLAY"O1 L4 S10 M600 EAGGFD"
2120 RETURN
2130 REM EXPLOSIE
2140 SOUND 0,127:SOUND 1,2:SOUND 7,5:SOUND 13
,0:SOUND 8,16
2150 RETURN
```

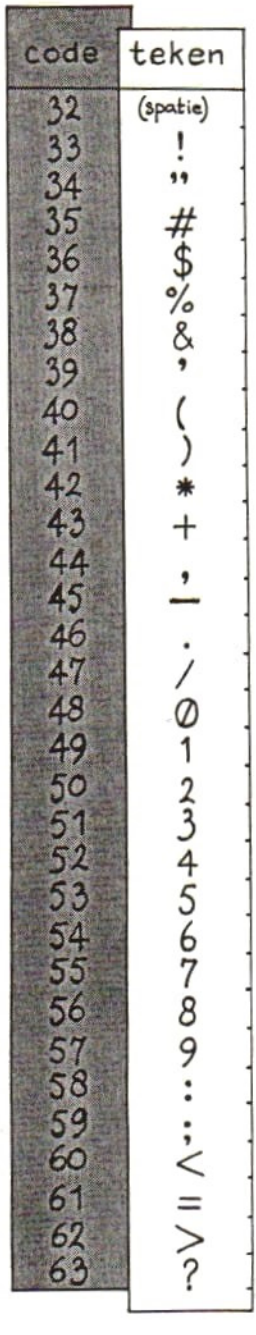
BIJLAGEN

Bijlage 1

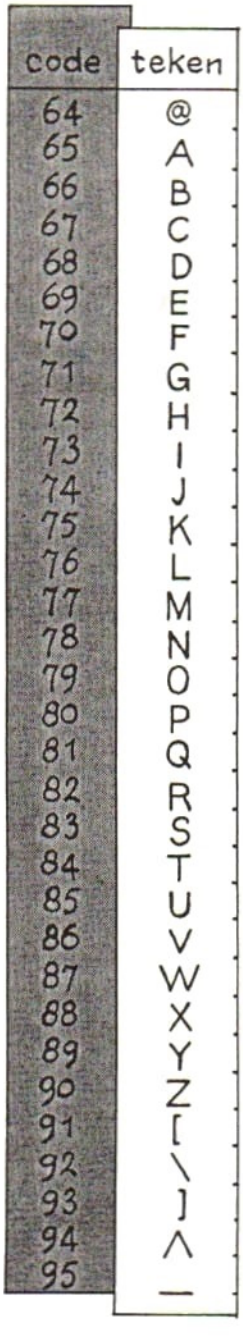
Complete karakterset



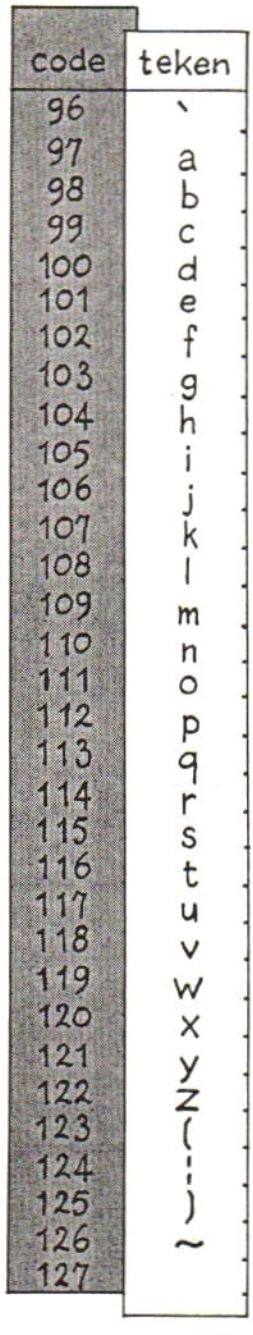
code	teken
0	(nul)
1	☺
2	☻
3	♥
4	♦
5	♣
6	♠
7	•
8	◘
9	○
10	◙
11	♂
12	♀
13	♪
14	♫
15	☼
16	►
17	◄
18	↕
19	‼
20	¶
21	§
22	▬
23	↨
24	↑
25	↓
26	→
27	←
28	∟
29	↔
30	▲
31	▼



code	teken
32	(spatie)
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?



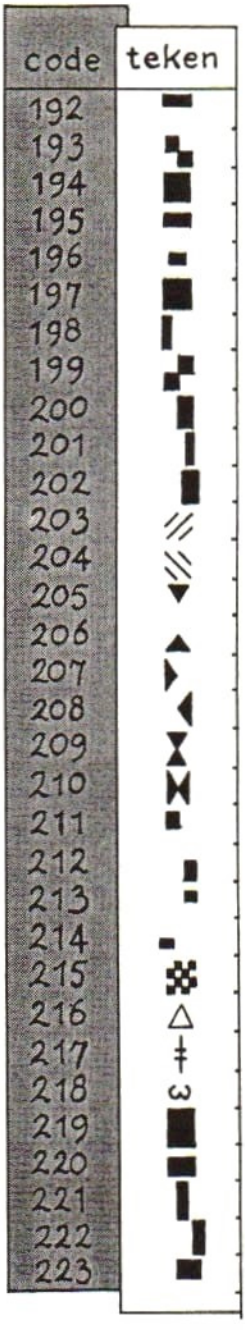
code	teken
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_



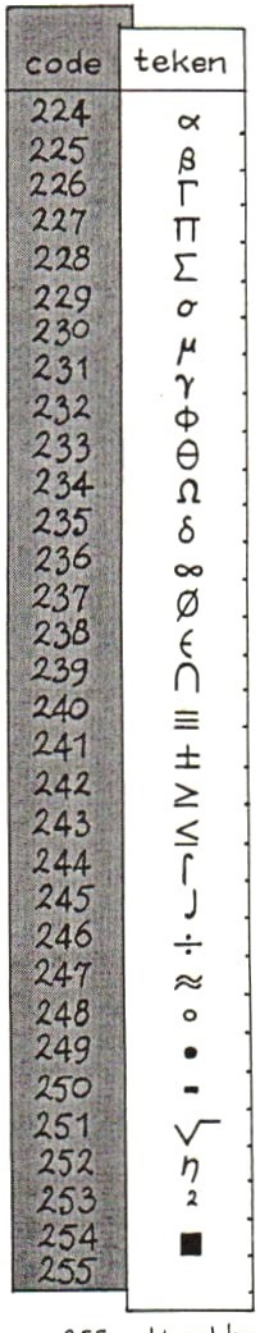
code	teken
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	

code	teken
128	Ç
129	ü
130	é
131	â
132	ä
133	à
134	å
135	ç
136	ê
137	ë
138	è
139	ï
140	î
141	ì
142	Ä
143	Å
144	É
145	æ
146	Æ
147	ô
148	ö
149	ò
150	û
151	ù
152	ÿ
153	Ö
154	Ü
155	¢
156	£
157	¥
158	₧
159	ƒ

code	teken
160	á
161	í
162	ó
163	ú
164	ñ
165	Ñ
166	ª
167	º
168	¿
169	⌐
170	¬
171	½
172	¼
173	¡
174	«
175	»
176	ã
177	ã
178	ĩ
179	ĩ
180	Õ
181	õ
182	Ũ
183	ũ
184	Ĳ
185	ĳ
186	¾
187	‚
188	◇
189	‰
190	¶
191	§



code	teken
192	
193	
194	
195	
196	
197	
198	
199	
200	
201	
202	
203	
204	
205	
206	
207	
208	
209	
210	
211	
212	
213	
214	
215	
216	
217	
218	
219	
220	
221	
222	
223	



code	teken
224	
225	
226	
227	
228	
229	
230	
231	
232	
233	
234	
235	
236	
237	
238	
239	
240	
241	
242	
243	
244	
245	
246	
247	
248	
249	
250	
251	
252	
253	
254	
255	

255: aktueel karakter
onder de cursor

Bijlage 2

Kleurcodes

code	kleur
0	transparant
1	zwart
2	groen
3	lichtgroen
4	donkerblauw
5	lichtblauw
6	donkerrood
7	hemelsblauw
8	rood
9	lichtrood
10	donkergeel
11	lichtgeel
12	donkergroen
13	magenta
14	grijs
15	wit

Bijlage 3

Foutmeldingen

Deze bijlage geeft een opsomming van de meest voorkomende foutmeldingen die de MSX kan geven, met daarbij de oorzaak van de fout.

Bad file name
- de ingevoerde bestandsnaam is onjuist
- de ingevoerde apparaatnaam is niet bruikbaar met een OPEN, SAVE of LOAD instructie

Can't continue
- na een onderbreking kan de verwerking van het programma niet worden hervat, aangezien er een wijziging is aangebracht
- het gevraagde programma bestaat niet
- in het programma was al een CONT-instructie gegeven

Device I/O error
- laden is niet mogelijk, er is iets met de recorder of met de drive
- de niveau-instelling van de recorder is onjuist
- er heeft een onderbreking plaatsgevonden voor het laden was afgelopen
- de omgeschreven I/O-poort is niet de juiste

Division by zero
- er wordt getracht door 0 te delen; dat is wiskundig gezien onmogelijk en dus ongeoorloofd
- er wordt getracht te delen door een variabele die niet is gedefinieerd

Extra ignored
- u hebt teveel data ingetypt als antwoord op een INPUT-instructie. Alleen de eerste data worden geaccepteerd.

File already open
- er wordt getracht een bestand te openen dat al geopend is

File not found
- wordt gegeven als u naar een bestand zoekt dat niet op band of schijf staat

File not OPEN
- in het uit te voeren bevel (PRINT#, INPUT#, enzovoort) is een be-

standsnummer gebruikt van een bestand dat nog niet is geopend met een OPEN-instructie

Illegal direct
- er wordt getracht rechtsreeks een instructie uit te voeren die alleen in een programma kan worden gebruikt, zoals de instructie DEF FN

Illegal function call
- in de gegeven instructie wordt een onjuiste waarde gebruikt
- de waarde van een functie ligt buiten het beschikbare bereik

Missing operand
- de benodigde parameter bij een instructie ontbreekt
- de benodigde parameters bij een instructie zijn niet compleet

NEXT without FOR
- er is een NEXT-instructie zonder bijbehorende FOR-instructie in een programma aanwezig
- bij de verwerking wordt door een GOTO-instructie verwezen naar een regel die zich binnen een FOR/NEXT-lus bevindt

Out of DATA
- tijdens de uitvoering van een READ-instructie blijkt dat er niet voldoende of zelfs geen gegevens aanwezig zijn

Out of memory
- het programma is te lang
- er zijn teveel lijstvariabelen gebruikt
- een lijstvariabele is te groot
- de totale structuur van de FOR/NEXT- of GOSUB RETURN-instructie is te lang

Out of string space
- het bereik dat voor lettertekens beschikbaar is, is overschreden
- het bereik dat met een CLEAR-instructie voor lettertekens beschikbaar is gemaakt, is onvoldoende groot

Overflow
- numerieke gegevens of het resultaat van een berekening vallen buiten het beschikbare bereik
- de parameter van een adres ligt buiten het beschikbare bereik

RETURN without GOSUB
- een RETURN-instructie heeft geen bijbehorende GOSUB-instructie

- er wordt door een GOTO-instructie verwezen naar een subroutine
- aangezien een programma niet is voorzien van een END-instructie, wordt automatisch doorgegaan naar een daarop volgende subroutine

Redimensioned array
- er wordt getracht elkaar overlappende lijstvariabelen te gebruiken met dezelfde naam
- er zijn ongedimensioneerde lijstvariabelen gebruikt, die pas daarna met een DIM-instructie werden gedefinieerd

String too long
- aan een rijvariabele is een waarde toegekend die langer is dan 255 lettertekens

Subscript out of range
- er wordt in een lijstvariabele een sub-indexnummer gebruikt dat het formaat zoals bepaald met een DIM-instructie te boven gaat
- een sub-index groter dan 11 wordt gebruikt in een lijstvariabele die niet is gedefinieerd met een DIM-instructie

Syntax error
- een gegeven instructie voldoet niet aan de grammaticale regels in BASIC

Type mismatch
- in een LET-, INPUT- of READ-instructie worden links en rechts van het '='teken ongelijksoortige eenheden (numerieke of rij-waarden) gebruikt
- er wordt getracht een rekenkundige of logische bewerking uit te voeren met een rij lettertekens
- de gegevens die in een functie worden gebruikt zijn niet van het juiste type

Undefined line number
- in een GOTO-, GOSUB- of RESUME-instructie wordt verwezen naar een niet bestaand regelnummer
- bij de uitvoering van een RENUM-instructie blijkt dat in een GOTO- of soortgelijk instructie een niet bestaand regelnummer is gebruikt

Undefined user function
- er wordt getracht een gebruikersfunctie aan te spreken die niet is gedefinieerd met een DEF FN-instructie

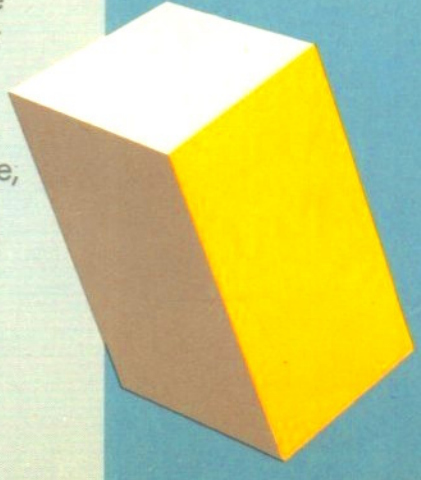
Verify error
- het programma op de cassette is niet gelijk aan het programma in het geheugen

MSX, de moderne standaard voor hardware en software, is in korte tijd een groot succes geworden. MSX-gebruikers hebben daardoor volop keus uit computers, randapparaten en programma's.
Uw MSX-computer is, mede dank zij deze standaardisering, een waardevol bezit. MSX-computers vallen onder meer op door hun voortreffelijke grafische eigenschappen. **Het MSX Instructieboek** helpt u ten volle profijt te trekken uit de rijke mogelijkheden die uw machine biedt.

Robert Young, die naam gemaakt heeft als auteur van computerboeken, legt in heldere taal uit hoe grafische programma's werken en hoe u zelf spannende spelletjes kunt programmeren. Een wereld van actie gaat voor u open. U leert gestructureerd programmeren, zodat u minder fouten zult maken en de *bugs* die onverhoopt toch nog voorkomen snel zelf kunt opsporen. Uw programma's zullen een professioneel aanzien hebben, dank zij de vele handige tips die **Het MSX Instructieboek** u geeft. U leert de drie kanalen van de MSX in te zetten, om uw programma's met geluidseffecten en melodieën nog interessanter te maken. Uw eigen creatieve inbreng wordt gestimuleerd, ook wat de *sprites* betreft.

Het MSX Instructieboek beschrijft diverse spannende actiespelen tot in detail. Start ze, geniet ervan, en... doe met de programmeertechniek en de kneepjes uw voordeel voor uw eigen programma's.

ISBN 90 6789 062 6



ADDISON-WESLEY