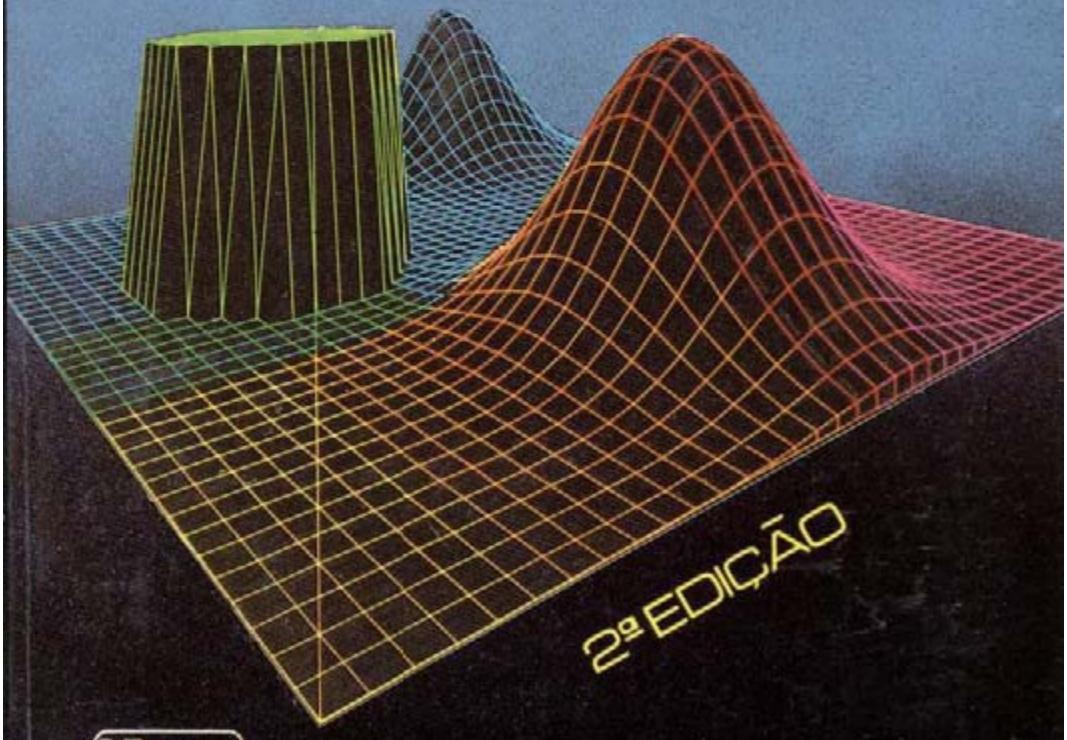
COLEÇÃO DE PROGRAMAS PARA MSX

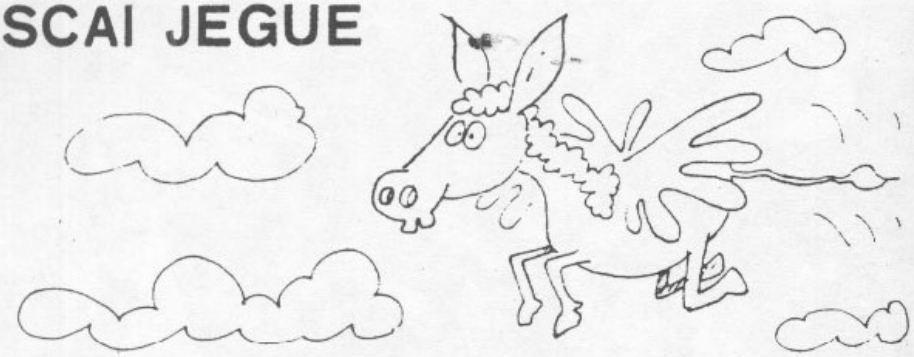


2ª EDIÇÃO

Editora Aleph

VOL. II

ISCAI JEGUE



INSTRUÇÕES

Um dos recursos mais potentes do MSX para elaboração de jogos de ação, é a possibilidade de se utilizar a SCREEN 1 com caracteres redefinidos. O programa apresentado a seguir é uma caricatura de um dos mais belos jogos que usam um recurso semelhante a esse: o SKY JAGAR (ou COLUMBIA). Vamos estudar passo a passo os estágios para sua elaboração.

Antes de mais nada é necessário uma idéia. É a partir dela que desenvolveremos o jogo. Vamos usar a mesma idéia do SKY JAGAR, isto é, a tela (SCREEN 1) deverá "rolar" para baixo e o jogador deverá controlar um jegue (uma nave, no original), desviando-se dos obstáculos (buracos, mata-burros e cobras) e, se possível, destruindo-os.

O controle do jegue será feito através das setas ao lado direito do teclado. Para eliminar obstáculos à sua frente, o jóquei que montar o jegue deverá usar a barra de espaços.

Os demais detalhes do jogo irão surgindo naturalmente durante a sua elaboração.

Para começar, vamos introduzir e analisar a rotina que faz a tela girar na vertical, de cima para baixo (figura 1.1). Na verdade, não é toda a tela que gira. A última linha, na parte inferior do vídeo, fica imóvel, reservada para apresentação de mensagens permanentes. Você entenderá o porquê disso mais adiante.

Figura 1.1 — Giro vertical da SCREEN 1

```
1000 '  Rotina para "ROLAR" a
1010 '  SCREEN 1  para baixo:
1020 CLEAR 200,&HC000
1030 FOR I = &HC000 TO &HC039
```

```
1040      READ X$
1050      POKE I,VAL("&H"+X$)
1060 NEXT I
1070 DATA F3,21,BF,1A,11,DF,1A,0E
1080 DATA 16,06,20,7D,D3,99,7C,D3
1090 DATA 99,F5,F1,DB,98,F5,7B,D3
1100 DATA 99,7A,F6,40,D3,99,F1,00
1110 DATA D3,98,2B,1B,10,E5,0D,20
1120 DATA E0,97,D3,99,3E,58,D3,99
1130 DATA 06,20,78,D3,98,00,10,FB
1140 DATA FB,C9
1150 DEFUSR0=&HC000
```

A parte principal dessa rotina é um programa em linguagem de máquina, cujos códigos estão inseridos em hexadecimal nas linhas DATAs de 1070 e 1140.

A linha 1020 reserva 200 bytes para variáveis "strings" e fixa o maior endereço que o programa em BASIC poderá usar.

O laço entre as linhas 1030 e 1060 lê os códigos hexadecimais e os insere a partir do endereço &HC000 da RAM. O endereço de execução dessa rotina é definido pela linha 1150. Cada vez que a rotina em linguagem de máquina é executada, ela gira a SCREEN 1 uma linha para baixo (com exceção da última linha!).

Para testar essa rotina, acrescente ao programa as linhas mostradas na figura 1.2.

Figura 1.2 — Teste do Giro Vertical da SCREEN 1

```
1160 '   Teste da rotina
1170 COLOR 9,4,7 : SCREEN 1 : KEY OFF
1180 LOCATE 32*RND(1),0
1190 PRINT "x"
1200 K=USR0(0)
1210 GOTO 1180
```

A linha 1170 define as cores de frente, fundo e borda; seleciona a SCREEN 1 (pois é nela que a rotina em linguagem de máquina produz o movimento vertical) e apaga as mensagens das teclas de funções na parte inferior da tela.

A linha 1180 "sorteia" uma coluna na parte superior da tela e a linha 1190 imprime um "X" nessa posição.

Logo a seguir, na linha 1200, a rotina em linguagem de máquina é executada através da instrução K=USR0(0), fazendo a tela toda "rolar" uma linha para baixo.

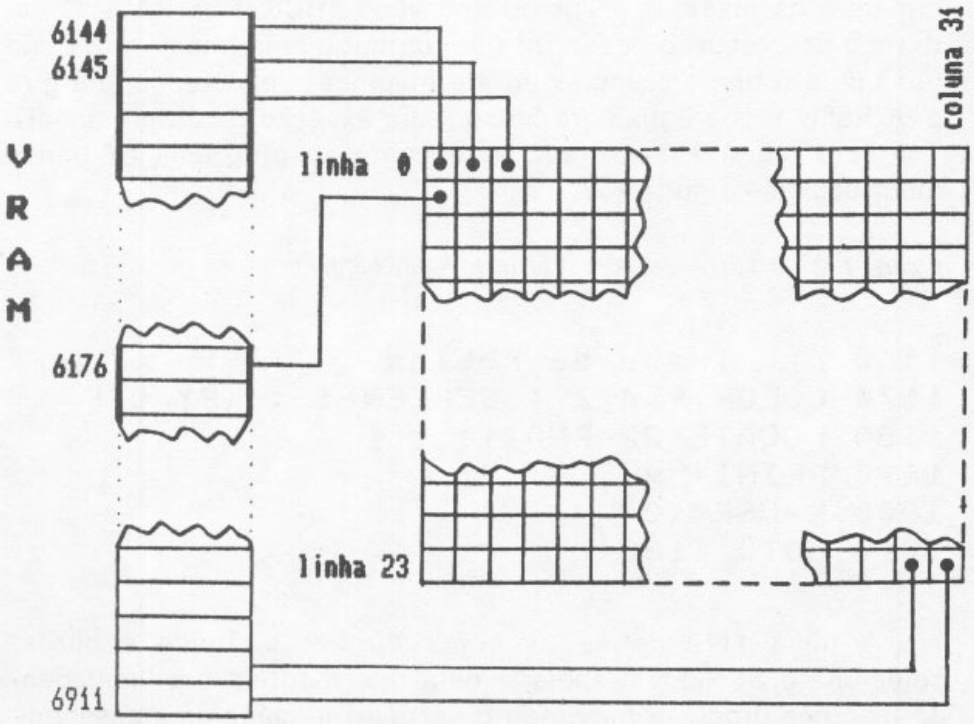
O comando GOTO na linha 1210 faz com que o processo seja repetido indefinidamente a partir da linha 1180.

Antes de prosseguirmos, vamos otimizar um pouco o que já fizemos.

A impressão de caracteres no vídeo pode ser feita por PRINT, como no teste da figura 1.2, ou por um VPOKE na VRAM. A segunda maneira é significativamente mais rápida que a primeira e, em jogos de ação, a velocidade é fundamental. Vamos, portanto, tentar usar o VPOKE ao invés do PRINT.

A imagem da SCREEN 1 é armazenada entre os endereços 6144 e 6911 (inclusive). O primeiro desses endereços (6144) contém o código do caractere apresentado no canto superior esquerdo da tela. Veja na figura 1.3 como a SCREEN 1 é mapeada e armazenada na VRAM.

Figura 1.3 — SCREEN 1



Portanto, para imprimir um caractere na linha superior da tela, devemos "vpokeá-lo" entre os endereços 6144 e 6144+31.

A sintaxe do VPOKE é:

VPOKE endereço, código do caractere

O código do caractere "x" é 120, portanto a instrução VPOKE 6144+32*RND(1),120 resolve o nosso problema.

Experimente eliminar as linhas 1188 e 1190 e substituí-las pela linha:

```
1180 VPOKE 6144+32*RND(1),120
```

Podemos usar o programa do jeito que ele está para produzir os obstáculos (representados pelo "x"). O caráčter "x", entretanto, não se parece nem um pouco com um obstáculo. Vamos substituí-lo pelo caractere "Ω" cujo código é 234.

Substitua a linha 1180 por:

```
1180 VPOKE 6144+32*RND(1),234
```

Agora, já temos a tela girando para baixo com os obstáculos. Só falta o jegue correndo.

Provisoriamente, podemos usar o caractere "Δ" cujo código é 216, para representar o jegue.

Insira a linha:

```
1190 VPOKE 6640,216
```

O jegue já está na tela deixando atrás de si um rastro de sujeira, que posteriormente iremos eliminar.

Vamos agora dar-lhe movimento.

Elimine as linhas de 1180 a 1210 e insira as linhas mostradas na figura 1.4.

Figura 1.4 — Movimentando o jegue pela tela

```
1180 DEFINT A-Z:X=6640
1190 VPOKE 6144+32*RND(1),234
1200 ' Movimentacao do jegue
1210 C=STICK(0)
1220 IF C>=6 AND C<=8 AND (X-1) MOD 32 T
HEN X=X-1
1230 IF (C=1 OR C=2 OR C=8) AND X>6176 T
HEN X=X-32
```

```
1240 IF C>=2 AND C<=4 AND (X+1) MOD 32 T
HEN X=X+1
1250 IF C>=4 AND C<=6 AND X<6848 THEN X=
X+32
1260 K=USR0(0)
1270 VPOKE X,216
1280 GOTO 1190
```

A linha 1180 define todas as variáveis usadas no programa como inteiras (isso aumenta um pouco a velocidade) e atribui à variável X o valor 6640 (posição inicial do jegue na VRAM).

A linha 1190, como já sabemos, gera os obstáculos no topo da tela.

A linha 1210 verifica as teclas das setas à direita do teclado através da função STICK(0).

As linhas seguintes alteram o valor da variável X, de acordo com a tecla de seta pressionada. Para irmos para a esquerda, basta subtrair uma unidade de X. Para irmos para a direita, basta acrescentar uma unidade a X. Para subirmos uma linha na SCREEN 1, temos que subtrair 32 unidades de X. Para descermos uma linha, devemos fazer o contrário, isto é, somar 32 unidades ao valor de X.

Os operadores lógicos OR e AND são usados para permitir movimentos nas diagonais e a função MOD é usada para evitar que o jegue saia da tela pelas laterais, desaparecendo de um lado e surgindo do outro.

A rotina para girar a tela é executada através da linha 1260 e o jegue é impresso pela linha 1270.

A linha 1280 começa tudo de novo a partir da impressão dos obstáculos no topo da tela.

Com isso, temos o jegue com movimento e a tela girando. Atrás do jegue, porém, ainda forma-se um rastro que deve ser apagado. Isso pode ser feito alterando-se a linha 1210 para:

```
1210 VPOKE X,32 : C=STICK(0)
```

O caractere 32 corresponde ao espaço em branco. Ele é "vpokeado" sobre o jegue antes que a tela gire, eliminando assim a formação do rastro.

Nosso próximo passo será checar quando o jegue colide com um obstáculo (o caractere de código 234). Se isso acon-

tecer, devemos desviar o programa para uma rotina que trate especificamente desse caso. Vamos fazer essa checagem alterando ou acrescentando as linhas mostradas na figura 1.5.

Figura 1.5 — Checagem de choque com obstáculo

```
1270 IF VPEEK(X)=234 THEN 1310
1280 VPOKE X,216
1290 GOTO 1190
1300 ' Rotina para obstaculo
1310 PRINT"  O jegue caiu numa vala!!!"
1320 FOR F=1 TO 3000:NEXT F
1330 GOTO 1170
```

A linha 1270 verifica se o caractere logo acima do jegue na tela é um obstáculo (caractere 234) e, em caso afirmativo, desvia o programa para a rotina que inicia na linha 1310. Caso contrário, a linha 1280 é executada e o jegue (caractere 216) é impresso.

A linha 1290 retoma a execução a partir da impressão dos obstáculos no topo da tela.

A rotina que começa na linha 1310 apenas imprime uma mensagem no vídeo ("O jegue caiu numa vala!!!"), espera alguns segundos e desvia a execução para a linha 1170, onde a tela é limpa pelo comando SCREEN 1.

Até aqui estamos ignorando propositalmente um dos recursos mais atraentes dos MSX, a produção de sons.

Vamos inicialmente produzir um som para o deslocamento do jegue. Faremos isso de uma forma não muito usual (mais um macete!... preste atenção!).

Além do PLAY e do SOUND, que usam o PSG, pode-se produzir um click (o mesmo do teclado!) através de um OUT (comando do BASIC MSX) numa das portas da PPI.

Esse circuito controla, entre outras coisas, o click do teclado e você poderá obter mais detalhes sobre seu funcionamento no capítulo 3 do livro "APROFUNDANDO-SE NO MSX".

Mude a linha 1260 para:

```
1260 OUT 170,255:K=USR(0):OUT 170,127
```

Isso deverá dar som ao movimento do jegue.

Agora vamos melhorar a rotina do choque com o obstáculo. Altere o programa introduzindo as linhas mostradas na figura 1.6.

Figura 1.6 — Rotina de choque melhorada

```
1300 ' Rotina para obstaculo
1310 VPOKE X,42:SOUND 7,247:SOUND 6,31
1320 SOUND 8,16:SOUND 12,64:SOUND 13,0
1330 FORF=1TO50:IFF/2=F\2THENVPOKEX,35
1340 FOR G=1TO20:NEXTG:VPOKEX,42:NEXTF
1350 GOTO 1190
```

Nessa rotina, há duas coisas em que você deve prestar atenção: o ruído e a imagem do choque. Ambos foram produzidos de uma forma bastante simples e o resultado é plenamente satisfatório.

Agora vamos dar uma arma ao jegue, isto é, ao seu controlador, para que ele possa destruir os obstáculos. Faremos isso introduzindo uma rotina para lançar raios laser mediante pressão da barra de espaços.

Para ativar a interrupção do programa quando a barra for pressionada, devemos usar as instruções STRIG(0) ON e ON STRIG GOSUB. Altere a linha 1160 conforme segue:

```
1160 STRIG(0) ON:ON STRIG GOSUB 1370
```

Depois, introduzia a rotina apresentada na figura 1.7.

Figura 1.7 — Raio LASER disparado

```
1360 ' Raio laser disparado
1370 VPOKE X,216
1380 FOR F=X-32 TO X-512 STEP -32
1390 VPOKE F,33
1400 IF VPEEK(F-32)<>32 THEN 1420
1410 NEXT F
1420 FOR G=X-32 TO F-32 STEP-32
1430 VPOKE G,32
1440 NEXT G
1450 VPOKE X,32
1460 RETURN
```

A linha 1370 coloca o jegue na tela.

O laço entre as linhas 1380 e 1410 lança em raio formado por uma sequência de caracteres "!" (código 33) a partir do jegue. Dentro do laço, a linha 1400 verifica se algum obstáculo foi atingido e, em caso positivo, desvia o programa para a linha 1420.

O laço entre as linhas 1420 e 1450 apenas apaga o raio lançado.

Finalmente, a linha 1460 retorna para o programa principal.

Com essa nova arma, o jegue ficou invencível! Um SUPER-ISCAI-JEGUE! O jogo, desse jeito, não tem graça. Vamos melhorar um pouco essa rotina, fazendo com que, ao errar um tiro, o disparador laser fique algum tempo sem funcionar.

Introduza e altere algumas linhas do programa conforme segue:

```
1200 CT=CT+1:IF CT=10 THEN FL=1:CT=0
1370 IFFL=0THENRETURNELSEVPOKEX,216
1410 NEXT F : FL=0 : CT=0
```

Essas linhas fazem com que o raio laser fique algum tempo desativado após um disparo ser dado sem atingir nenhum obstáculo.

Ainda assim, o jogo está um pouco sem graça, pois não há limite para a quantidade de colisões que o jegue pode ter com os obstáculos. Vamos limitar esse número em três, acrescentando ao programa as alterações mostradas na figura 1.8.

Figura 1.8 — Finalização do jogo

```
1180 DEFINT A-Z:X=6640:N=3
1190 VPOKE 6144+32*RND(1),234:LOCATE 0,2
3:PRINT"CHANCES:";STRING$(N,216);" ";
1350 N=N-1:IF N>0 THEN 1190 ELSE 1470
1470 ' rotina de finalizacao
1480 COLOR 14,1,13 : SCREEN 1
1490 PRINT" O JEGUE MORREU !!!"
1500 PRINT" Para obter outro JEGUE"
1510 PRINT" digite a tecla RETURN."
1520 INPUT X$
1530 RUN
```

Agora, podemos dizer que já temos um jogo pronto. A partir daqui podemos começar a enfeitá-lo.

Vamos fazer com que a cada obstáculo destruído o jegue ganhe um ponto e, a cada 100 pontos acumulados, faremos com que ele ganhe uma nova chance. Para isso basta introduzir as alterações apresentadas na figura 1.9.

Figura 1.9 — Contagem de pontos e bônus

```
1180 DEFINT A-Z:X=6640:N=3:PT=0
1190 VPOKE 6144+32*RND(1),234:LOCATE 0,2
3:PRINT"CHANCES:";STRING$(N,216);STRING$
(4-N,32);" PONTOS:";USING"###";PT;
1270 IF VPEEK(X)=234 THEN 1300
1300 PT=0
1400 IF VPEEK(F-32)<>32 THEN PT=PT+1:GOT
O 1420
1440 NEXT G:IF PT=100 THEN PT=0:N=N+1:IF
 N=5 THEN N=4
```

Mesmo com todos os caracteres impressos, a tela do jogo ainda está um pouco pobre, pois só tem três cores. Na SCREEN 1 existe a possibilidade de definirmos as cores de cada grupo de 8 caracteres seguidos. O MSX dispõe de 256 caracteres numerados de 0 a 255. Portanto, temos 32 grupos de 8 caracteres cada. O primeiro grupo começa com o caractere de código 0 e termina com o caractere de código 7. O último grupo é o dos caracteres com códigos entre 248 e 255. Para cada grupo, podemos definir uma cor de frente e uma cor de fundo diferentes.

A região da VRAM que armazena as cores de frente e fundo desses 32 grupos de caracteres fica entre os endereços 8192 e 8223. Na tabela da figura 1.10 podemos observar os trinta e dois "octetos" de caracteres e os respectivos endereços na VRAM onde suas cores de frente e fundo estão definidas.

Vamos alterar a cor do jegue. O código de seu caractere é 216, portanto o endereço da VRAM onde suas cores são definidas é o 8219.

O byte 8219 da VRAM, como qualquer outro, possui 8 bits. Cada bit pode valer 0 ou 1 (figura 1.11).

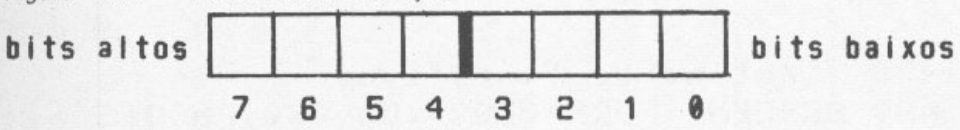
Os 4 bits mais baixos desse byte (os da direita) devem armazenar a cor de fundo e os 4 bits mais altos (os da esquerda) devem armazenar a cor de frente. Note que cada grupo de 4 bits pode armazenar um número entre 0 e 15 (em binário). Por exemplo, se inserirmos o número binário &B10110100 no endereço 8219 da VRAM o jegue será desenhado em amarelo sobre fundo azul, pois &B1011=11 (cor amarela) corresponde

à cor de frente e &B0100=4 (cor azul) representa a cor de fundo!

Figura 1.10 — Tabela de cores da SCREEN 1

Endereço	Caracteres alterados							
8192	0	1	2	3	4	5	6	7
8193	8	9	10	11	12	13	14	15
8194	16	17	18	19	20	21	22	23
8195	24	25	26	27	28	29	30	31
8196	32	33	34	35	36	37	38	39
8197	40	41	42	43	44	45	46	47
8198	48	49	50	51	52	53	54	55
8199	56	57	58	59	60	61	62	63
8200	64	65	66	67	68	69	70	71
8201	72	73	74	75	76	77	78	79
8202	80	81	82	83	84	85	86	87
8203	88	89	90	91	92	93	94	95
8204	96	97	98	99	100	101	102	103
8205	104	105	106	107	108	109	110	111
8206	112	113	114	115	116	117	118	119
8207	120	121	122	123	124	125	126	127
8208	128	129	130	131	132	133	134	135
8209	136	137	138	139	140	141	142	143
8210	144	145	146	147	148	149	150	151
8211	152	153	154	155	156	157	158	159
8212	160	161	162	163	164	165	166	167
8213	168	169	170	171	172	173	174	175
8214	176	177	178	179	180	181	182	183
8215	184	185	186	187	188	189	190	191
8216	192	193	194	195	196	197	198	199
8217	200	201	202	203	204	205	206	207
8218	208	209	210	211	212	213	214	215
8219	216	217	218	219	220	221	222	223
8220	224	225	226	227	228	229	230	231
8221	232	233	234	235	236	237	238	239
8222	240	241	242	243	244	245	246	247
8223	248	249	250	251	252	253	254	255

Figura 1.11 — Estrutura de um byte



Introduza a linha a seguir no programa.

```
1175 VPOKE 8219,&B10110100
```

Ela servirá para destacarmos o jegue dos demais caracteres impressos.

Aproveite para melhorar um pouco a linha 1170, deixando-a como segue:

```
1170 COLOR9,4,7:SCREEN1,,0:KEYOFF
```

Dessa forma eliminamos o click de teclado.

Vamos agora colorir mais ainda o jogo. Inicialmente podemos mudar a cor do raio laser de vermelho para cinza (cor 14=&B1110). O caractere usado para produzir o raio é o "!" cujo código é 33. Olhando a tabela da figura 1.10 vemos que o endereço a ser alterado é o 8196, portanto basta introduzir a linha 1176 mostrada a seguir para fazê-lo ficar cinza e azul.

```
1176 VPOKE 8196,&B11100100
```

Agora vamos colorir todas as mensagens mostradas na parte inferior da tela, deixando-as brancas. Os caracteres usados são as letras A,C,E,H,N,A,P,S,T e o sinal ":". Introduza a linha a seguir:

```
1177 FOR F=8198 TO 8202 : VPOKE F,&B1111
0100 : NEXT F
```

Uma outra implementação que podemos fazer é produzir um som toda vez que o raio laser for disparado. A linha adiante se encarrega disso.

```
1375 SOUND7,52:SOUND6,8:SOUND8,16:SOUND9
,16:SOUND0,99:SOUND1,0:SOUND2,199:SOUND3
,0:SOUND12,3:SOUND13,4
```

Agora vamos mexer mais um pouco no final do jogo. Tocaremos uma marcha fúnebre para o jegue morto. Na figura 1.12 vemos as linhas a serem acrescidas à rotina de finalização.

Figura 1.12 — A Morte do Jegue com Pompas

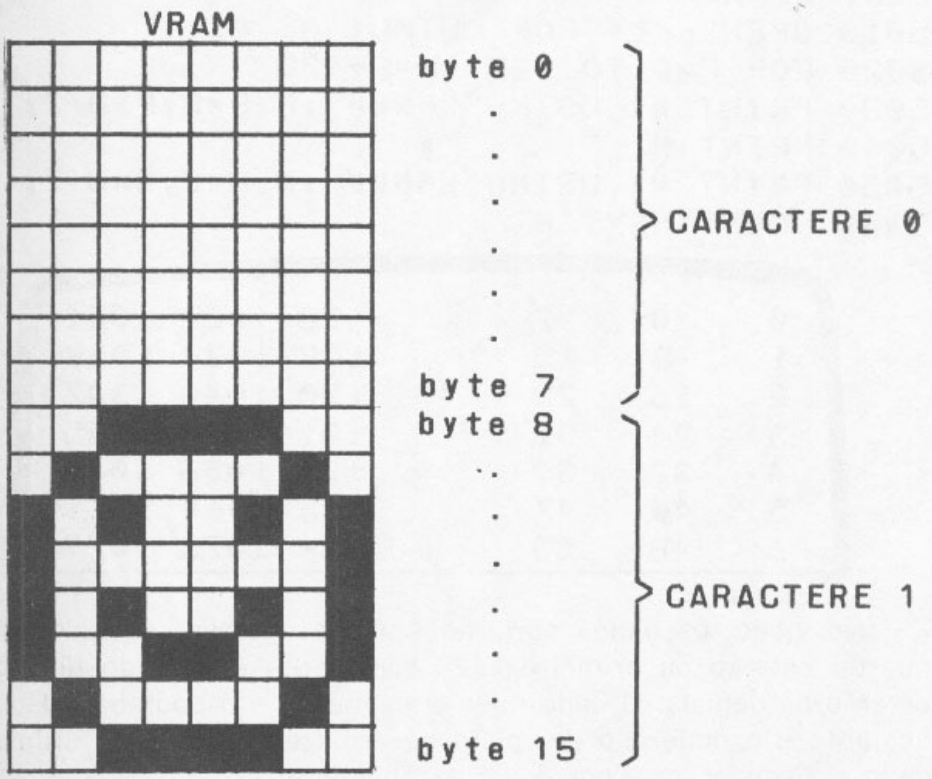
```
1470 ' rotina de finalizacao
1480 SCREEN1:FORF=0TO99:COLOR0,FMOD15
```

```
1490 NEXTF:COLOR6,1,13:SOUND7,56
1495 LOCATE5,5:PRINT"O JEGUE MORREU !"
1500 PRINT" Para ressuscitar o JEGUE"
1510 PRINT"digite a barra de espacos."
1520 PLAY"t110m100s0v15","t110v15"
1530 PLAY"o1bbbbo2dc#c#o1bba#br8","o3bbb
bo4dc#c#o3bba#br8"
1540 STRIG(0)OFF
1550 IFSTRIG(0)=0THEN1550ELSERUN
```

Como você deve estar notando, podemos fazer mil coisas para melhorar cada vez mais o jogo. Antes de deixarmos as alterações por sua conta, vamos fazer mais uma: redefiniremos os caracteres do jegue e do obstáculo!

Assim que a instrução SCREEN 1 é executada, o MSX copia todos os 256 caracteres da ROM para uma região da VRAM entre os endereços 0 e 2047. Cada caractere é representado por oito bytes em sequência (figura 1.13).

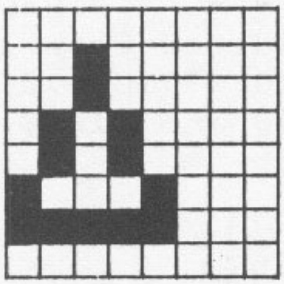
Figura 1.13 — Caracteres na VRAM



Por exemplo, o caractere "Δ" é armazenado como mostra a figura 1.14.

Figura 1.14 — O caractere"Δ" na VRAM

```
1728 ------> 00000000 ---->
1729 ------> 00100000 ---->
1730 ------> 00100000 ---->
1731 ------> 01010000 ---->
1732 ------> 01010000 ---->
1733 ------> 10001000 ---->
1734 ------> 11111000 ---->
1735 ------> 00000000 ---->
```



O programa da figura 1.15 apresenta no vídeo os endereços iniciais e finais de armazenamento dos 256 caracteres na VRAM quando se usa a SCREEN 1.

Figura 1.15 — Endereços de caracteres na VRAM

```
5000 SCREEN 0:KEY OFF
5010 OPEN"crt:"FOR OUTPUT AS #1
5020 FOR F=0 TO 127 :H=F+128
5030 PRINT #1,USING"######";F;F*8;F*8+7;
5040 PRINT #1,"      ";
5050 PRINT #1,USING"######";H;H*8;H*8+7;
5060 PRINT:NEXT F
```

```
 0     0     7        128 1024 1031
 1     8    15        129 1032 1039
 2    16    23        130 1040 1047
 3    24    31        131 1048 1055
 4    32    39        132 1056 1063
 5    40    47        133 1064 1071
 6    48    55        134 1072 1079
```

No vídeo, os dados surgirão em duas partes: ao lado esquerdo estarão os primeiros 128 caracteres e ao lado direito estarão os demais. O dado mais à esquerda é o código, o dado seguinte é o endereço de início de armazenamento e o último dado é o endereço final de armazenamento.

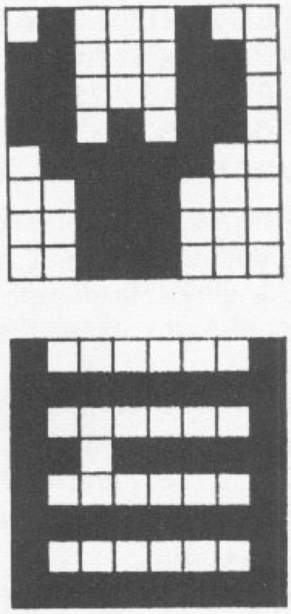
Para redefinir um caractere basta alterar os bytes que o definem na VRAM.

Vamos alterar o jegue (código 216) e os obstáculos (código 234). Usando o programa da figura 1.15 vemos que eles são armazenados, respectivamente, nos endereços 1728-1735 e 1872-1879. Vamos alterá-los, deixando-os como na figura 1.16.

Figura 1.16 — O jegue e os obstáculos redefinidos

```
1728 => 01000100 =>
1729 => 11000110 =>
1730 => 11000110 =>
1731 => 11010110 =>
1732 => 01111100 =>
1733 => 00111000 =>
1734 => 00111000 =>
1735 => 00111000 =>

1872 => 10000001 =>
1873 => 11111111 =>
1874 => 10000001 =>
1875 => 11011111 =>
1876 => 10000001 =>
1877 => 11111111 =>
1878 => 10000001 =>
1879 => 11111111 =>
```



Introduza as linhas mostradas na figura 1.17 para redefinir os caracteres.

Figura 1.17 — Rotina de redefinição

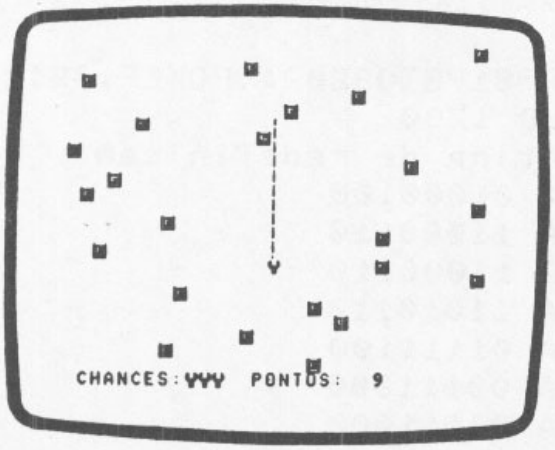
```
1177 FORF=8198TO8202:VPOKEF,&B11110100:N
EXTF:GOSUB 1730
1560 ' Rotina de redefinicao
1570 DATA 01000100
1580 DATA 11000110
1590 DATA 11000110
1600 DATA 11010110
1610 DATA 01111100
1620 DATA 00111000
1630 DATA 00111000
```

```
1640 DATA 00111000
1650 DATA 10000001
1660 DATA 11111111
1670 DATA 10000001
1680 DATA 11011111
1690 DATA 10000001
1700 DATA 11111111
1710 DATA 10000001
1720 DATA 11111111
1730 RESTORE 1570:FOR F=0 TO 7
1740 READ X$:A=VAL("&b"+X$)
1750 VPOKE 1728+F,A : NEXT F
1760 FORF=0TO7:READX$:A=VAL("&B"+X$)
1770 VPOKE1872+F,A:NEXTF:RETURN
```

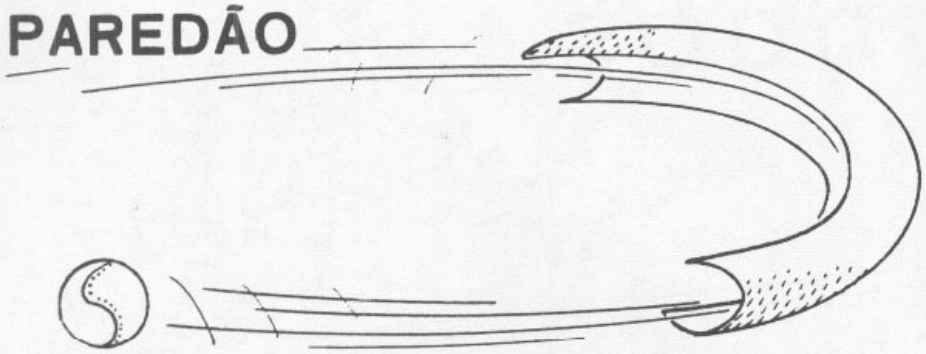
A partir daqui as alterações ficam por sua conta. Aqui vão algumas sugestões:

0) Crie uma tela de apresentação para o jogo!
1) Redefina os caracteres "espaço em branco" cujo código é 32.
2) Redefina o caractere "!", cujo código é 33.
3) Gere ao menos 3 tipos diferentes de obstáculos (não apenas o caractere 234) e faça com que um deles tenha que ser necessariamente destruído antes de chegar ao fim da tela.
4) Transforme o JEGUE num jato e os obstáculos em naves inimigas.
5) Não esqueça de gravar o programa !!!"
6) Jogue !!!

No livro "Programação Avançada em MSX" ensinamos outros truques que se pode fazer com as telas no MSX.



PAREDÃO



INSTRUÇÕES

Este programa é uma versão em BASIC de um dos primeiros "video-games" produzidos comercialmente. Obviamente, a velocidade do jogo é bastante lenta quando o comparamos com versões feitas em linguagem de máquina, no entanto, a estrutura do programa é o que mais nos importa no momento e compreendê-la poderá ser de grande utilidade quando você for desenvolver novos programas.

O jogo é bastante simples: você controla uma raquete com as teclas ▲ e ▼ e deve impedir que uma bola de tênis saia pelo lado direito da tela (quadra). Há um total de 6 bolas, ao término das quais o jogo acaba.

Figura 2.1 — Programa PAREDÃO

```
1000 REM  JOGO PAREDAO
1010 REM  Rubens & Renato (FEV/86)
1020 REM ------------------------------
1030 CQ=3 : CP=15 : CR=6
1040 CB=9 : CN=12
1050 COLOR CP,CQ,CQ
1060 SCREEN 3,,0
1070 OPEN "GRP:" FOR OUTPUT AS #1
1080 FOR F=1 TO 5
1090   READ A$
1100   B$=B$+CHR$(VAL("&h"+A$))
1110 NEXT F
1120 DATA 70,c8,e8,f8,70
1130 SPRITE$(1)=B$
1140 DRAW"BM255,0L255D191R255"
1150 COLOR CR
```

```
1160 PSET(230,80),CR : DRAW"D16"
1170 XR=230 : YR=80
1180 PT=0
1190 FOR NB=6 TO 1 STEP -1
1200  XB=8 : YB=INT(180*RND(-TIME))+4
1210  DX=4 : DY=RND(-TIME)*9-4
1220  COLOR CQ
1230  PSET (40,40),CQ : PRINT #1,NB+1
1240  COLOR CN
1250  PSET (40,40),CQ : PRINT #1,NB
1260  IF STICK(0)=1 AND YR>7 THEN PSET(X
R,YR+16),CQ : YR=YR-4 : PSET(XR,YR),CR
1270  IF STICK(0)= 5 AND YR<168 THEN PSE
T(XR,YR),CQ : YR=YR+4 : PSET(XR,YR+16),C
R
1280  XB=XB+DX : YB=YB+DY
1290  IF XB>260 THEN 1510
1300  IF XB<4 THEN DX=-DX : XB=XB+DX
1310  IF YB<4 OR YB>183 THEN DY=-DY:YB=Y
B+DY
1320  IF XB>XR-4 AND XB<XR+1 AND YB>YR-4
 AND YB<YR+16 THEN 1350
1330  PUT SPRITE 1,(XB,YB),CB,1
1340  GOTO 1260
1350  SOUND 0,250
1360  SOUND 1,1
1370  SOUND 2,250
1380  SOUND 3,0
1390  SOUND 7,56
1400  SOUND 8,16
1410  SOUND 9,16
1420  SOUND 11,255
1430  SOUND 12,2
1440  SOUND 13,0
1450  IF YB<YR+4 OR YB>YR+16 THEN NX=4 :
 NY=4 : GOTO 1480
1460  IF YB<YR+8 OR YB>YR+12 THEN NX=4 :
 NY=2 : GOTO 1480
1470  NX=4 : NY=1
1480  DX=-NX : DY=SGN(DY)*NY
1490  PT=PT+1
```

```
1500  GOTO 1280
1510 NEXT NB
1520 LINE(4,4)-(251,187),4,BF
1530 PSET(20,20),4 : PRINT #1,"PONTOS: "
;PT
1540 IF STRIG(0)=0 THEN 1540
1550 CLOSE : RESTORE : RUN
```

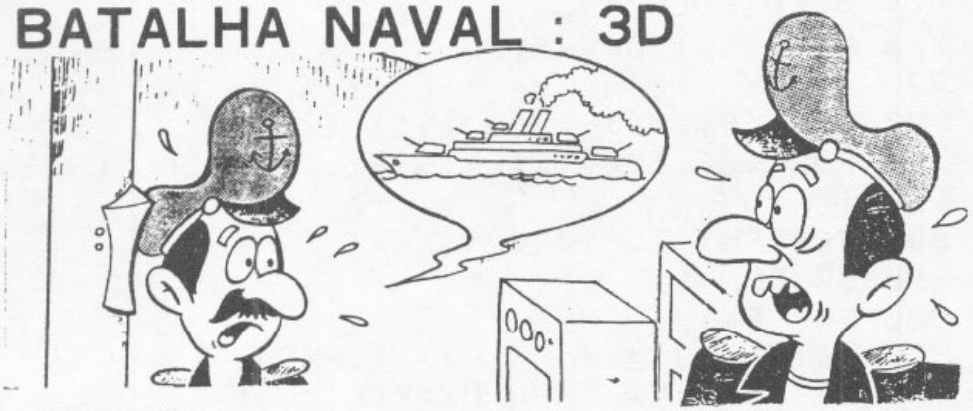
ANÁLISE

Os nomes das variáveis indicam as funções a que elas se prestam.

CQ = cor da quadra
CP = cor dos pontos
CB = cor da bola
CN = cor dos números
PT = pontos
NB = número de bolas
XB = coordenada X da bola
YB = coordenada Y da bola
DX = incremento (Delta) na coordenada X da bola
DY = incremento (Delta) na coordenada Y da bola
XR = coordenada X da raquete
YR = coordenada Y da raquete
NX = número do incremento de X da bola
NY = número do incremento de Y da bola

A bola é um "sprite" definido entre as linhas 1110 e 1170.
As linhas 1300 e 1310 movem a raquete.
Para iniciar o jogo, pressione a barra de espaços.
Um exercício interessante seria tentar fazer este mesmo jogo usando a SCREEN 1 ao invés da SCREEN 3 e nesse caso, certamente a velocidade do jogo seria bem maior!
Se você está disposto, aqui fica o desafio!

BATALHA NAVAL : 3D



INSTRUÇÕES

Esse jogo não é o tradicional Batalha Naval, já bastante conhecido, onde dois competidores posicionam suas esquadras em um campo quadriculado e cada um tenta afundar os navios do outro.

A finalidade desse jogo é também afundar navios porém, apenas aqueles posicionados pelo computador. Trata-se de um jogo de estratégia, memória e percepção visual.

O computador escolhe aleatoriamente cinco posições e nelas coloca os seus navios. Sua missão é tentar afundá-los. Para isso, eles serão mostrados assim que a tela estiver pronta e você pressionar uma tecla qualquer. Os navios serão, no entanto, mostrados em perspectiva e durante alguns segundos apenas. Você deve, neste intervalo de tempo, avaliar as coordenadas dos navios e memorizá-las pois eles sumirão do seu campo visual. Quando isso acontecer, você terá dez tiros para tentar afundá-los.

Os tiros devem ser dados informando ao computador as coordenadas da posição que você pretende atingir. Informe primeiro a coordenada horizontal (uma letra de A a J) e depois a vertical (um número de 0 a 9) e finalmente confirme o tiro pressionando a tecla "S".

Se o computador não aceitar as coordenadas que você entrou, pressione a tecla "CAPS LOCK" uma única vez e tente de novo.

Caso você erre o tiro será pintada de azul a posição correspondente no mapa quadriculado. Caso acerte, ela será pintada de vermelho e serão computados os pontos. O número de pontos que você ganha por afundar um navio é tanto maior quanto mais navios já tiver afundado e tanto maior quanto mais avançada for a fase em que você se encontra.

Para passar para uma nova fase, todos os cinco navios devem

ser atingidos. Nesse caso, outras posições serão sorteadas para cinco novos navios e o jogo recomeça com a sua pontuação mantida e mais dez tiros.

Quanto mais avançada for a fase, maior será a dificuldade pois menor será o tempo em que os navios serão mostrados.

Caso você dê os dez tiros, numa fase qualquer, e não consiga afundar todos os navios, o jogo termina.

Figura 9.1 — Programa BATALHA NAVAL - 3D

```
100 ' BATALHA NAVAL
110 ' LUIZ TARCISIO DE CARVALHO JR
120 OPEN"grp:"FOROUTPUTAS#1
130 DIMM(10,10)
140 SCREEN2:HI=0:SC=0:Q=0:W=0
150 TA=10: COLOR15,1,1:CLS
160 PSET(206,100),1:COLOR8:PRINT#1,"RECO
RD"
170 PSET(198,110),1:COLOR15:PRINT#1,HI
180 PSET(206,130),1:COLOR8:PRINT#1,"PONT
OS"
190 PSET(198,140),1:COLOR15:PRINT#1,SC
200 PSET(206,160),1:COLOR8:PRINT#1,"FASE
"
210 PSET(198,170),1:COLOR15:PRINT#1,Q+1
220 FORX=152TO232STEP8
230 LINE(X,0)-(X,80),11:NEXTX
240 FORY=0TO80STEP8
250 LINE(152,Y)-(232,Y),11:NEXTY
260 COLOR4: FORX=152TO224STEP8
270 PSET(X+2,85),1:PRINT#1,CHR$((X-152)/
8+65):NEXTX
280 FORY=0TO72STEP8
290 PSET(233,Y),1:PRINT#1,9-(Y/8):NEXTY
300 LINE(50,90)-(150,90),5
310 LINE-(200,190),5
320 LINE-(0,190),5
330 LINE-(50,90),5
340 PAINT(100,189),5
350 COLOR15,1
```

```
360 SPRITE$(1)=CHR$(&H40)+CHR$(&H6)+CHR$
(&H30)+CHR$(&H12)+CHR$(&H58)+CHR$(&H59)+
CHR$(&HFB)+CHR$(&H7B)
370 SPRITE$(2)=CHR$(&HC)+CHR$(&H1E)+CHR$
(&HFF)+CHR$(&H7F)+CHR$(&H0)+CHR$(&H0)+CH
R$(&H0)+CHR$(&H0)
380 GOSUB840
390 FORI=0TO4:X=5+5*Y(I)+X(I)*(19-Y(I)):
Y=180-Y(I)*10:PUTSPRITEI,(X,Y),15,2:NEXT
400 TIME=0:TI=Q*300:IFTI>1200THENTI=1000
410 IFTIME<(1200-TI)THENGOTO410
420 FORI=0TO4:PUTSPRITEI,(0,0),0,2:NEXT
430 GOSUB500:GOSUB450:GOSUB750
440 IFTA=0THENGOTO930ELSEGOTO430
450 TA=TA-1:GOSUB690
460 SOUND6,31:SOUND7,199:SOUND8,16:SOUND
9,16:SOUND10,16:SOUND12,20:SOUND13,0
470 X=5+5*KY+KX*(19-KY):Y=180-KY*10:PUTS
PRITE0,(X,Y),15,1:FORI=1TO200:NEXT
480 PUTSPRITE0,(X,Y),0,1
490 RETURN
500 LINE(30,12)-(140,41),1,BF
510 PSET(30,12),1:PRINT#1,"X ?(A-J) ";
520 GOSUB670
530 K=ASC(A$):IFK>64ANDK<75THEN550
540 GOTO520
550 KX=K-65:PRINT#1,A$
560 PSET(30,20),1:PRINT#1,"Y ?(0-9)";
570 GOSUB670
580 K=ASC(A$):IFK>47ANDK<58THEN600
590 GOTO570
600 KY=K-48:PRINT#1,KY
610 PSET(30,33),1:PRINT#1,"ATIRA ?(S/N)"
;
620 GOSUB670
630 PRINT#1,A$:IFA$<>"S"THEN650
640 RETURN
650 LINE(30,12)-(140,41),1,BF
660 GOTO500
670 A$=INKEY$:IFA$=""THEN670
680 RETURN
```

```
690 LINE(20,55)-(130,63),1,BF
700 IFTA=0THENRETURN
710 PSET(30,55),1
720 FORI=1TOTA
730 PRINT#1,CHR$(1)+CHR$(79);:NEXTI
740 RETURN
750 X1=KX*8+152:Y1=72-KY*8
760 LINE(X1,Y1)-(X1+7,Y1+7),7,BF
770 IFM(KX,KY)=1THEN790
780 COLOR15,1:RETURN
790 M(KX,KY)=0:W=W+1:SC=SC+10*(W+Q)
800 LINE(190,140)-(255,150),1,BF
810 LINE(X1,Y1)-(X1+7,Y1+7),8,BF
820 PSET(198,140),1:PRINT#1,SC
830 COLOR15,1:GOTO780
840 PSET(10,12),1:PRINT#1,"APERTE UMA TE
CLA"
850 IFINKEY$=""THEN850
860 LINE(10,12)-(140,20),1,BF
870 Z=INT(RND(-TIME)*60000!)+1
880 FORI=0TO4:X(I)=INT(RND(Z)*10):Y(I)=I
NT(RND(Z)*10):M(X(I),Y(I))=1:NEXT
890 FORI=0TO3
900 FORJ=I+1TO4
910 IFX(I)=X(J)ANDY(I)=Y(J)THEN870
920 NEXTJ,I:RETURN
930 IFW=5THENW=0:Q=Q+1:GOTO150
940 LINE(50,100)-(150,150),1,BF
950 PSET(60,110),1:PRINT#1,"NOVO JOGO?"
960 PSET(70,130),1:PRINT#1,"S ou N"
970 GOSUB670
980 IFSC>HITHENHI=SC
990 IFA$="S"THENERASEM:SC=0:W=0:Q=0:GOTO
150
```

ANÁLISE

O programa é bem simples quanto à sua lógica e seu detalhamento é o seguinte:

PROGRAMA PRINCIPAL

Linha 120: Abre um arquivo na tela de alta resolução para que possamos escrever textos nela.

Linha 130: dimensiona a matriz M(10,10) que terá cada elemento igual a 1, se existir um navio na posição correspondente a ele, e igual a 0, caso contrário.

Linha 140: seleciona a tela de alta resolução e "zera" as variáveis HI (record), SC (pontos), Q (fase) e W (número de navios afundados).

Linha 150: atribui o valor 10 à variável TA (número de tiros).

Linha 160 a 340: constroem a tela.

Linha 360: define o "sprite" correspondente ao desenho da explosão.

Linha 370: define o "sprite" correspondente ao desenho do navio.

Linha 380: "chama" a sub-rotina 840/920.

Linha 390: calcula as posições dos navios no plano em perspectiva e coloca os "sprites" dos navios.

Linhas 400 e 410: estabelece uma espera tanto menor quanto maior for o número da fase (Q).

Linha 420: "apaga" os "sprites' dos navios.

Linha 430: "chama" as sub-rotinas 500/680, 450/490 e 750/830.

Linha 440: se acabarem os tiros desvia para a linha 930, caso contrário volta a executar a linha 430 ("chamada" das sub-rotinas).

Linha 930 a 990: testam se os cinco navios foram afundados. Caso afirmativo, "zera" W (número de navios afundados), incrementa Q (fase) e desvia a execução do programa para a linha 150. Caso contrário, questiona-se a respeito do desejo de um novo jogo.

SUB-ROTINAS

450/490: dá o tiro, "chama" a sub-rotina 690/740, diminui a variável TA em uma unidade, produz o som do tiro, coloca e retira o "sprite" da explosão nas coordenadas do plano em perspectiva.

500/680: permite a entrada das coordenadas de cada tiro.

690/740: imprime os desenhos correspondentes aos tiros restantes.

750/830: pinta no mapa quadriculado a posição correspondente ao tiro dado. Se houver um navio nessa posição (M(KX,KY) =1), pinta-a de vermelho incrementa e imprime os pontos. Note que o incremento é tanto maior quanto maiores forem a fase e o número de navios já afundados.

840/920: espera a pressão de uma tecla para tornar aleatório o valor da variável TIME. Sorteia as cinco posições correspondentes aos navios e atribui o valor 1 aos respectivos elementos da matriz M(10,10). Verifica também se não houve repetição nas posições sorteadas.

vitacional gerado ao longo do plano XY, ou seja, impomos a coordenada Z como sendo zero. Desta forma, temos:

$$g = g\,(d \times d)$$

Portanto:

$$g = g\,(X \times X + Y \times Y)$$

Sendo g uma função de duas variáveis, podemos então usar o plano XY para mapear as posições ao redor de M, e um eixo ortogonal Z para indicar o campo gravitacional local.

O programa inicia pedindo o valor da massa que gerará o campo gravitacional. Logo após, ele já começa a traçar o gráfico correspondente. Para parar a execução em qualquer instante, tecle CONTROL+STOP.

Experimente valores de massa entre 50 e 5000 e veja que efeitos diferentes podem ocorrer. O que acontece quando a massa é zero?

Figura 13.1 — Programa CAMPO GRAVITACIONAL

```
100 ' Campo Gravitacional
110 ' Milton Maldonado Jr.
120 '
130 INPUT"Qual a massa";M:IFM<0THEN130
140 COLOR 15,4,4:SCREEN 2:DIM A%(255):FO
R X=0 TO 255:A%(X)=200:NEXT X:BEEP
150 D=0:FOR Y=0 TO 110 STEP5:Y1=Y
160 FOR X=0 TO 130:X1=X:GOSUB 200:NEXT X
:D=D+.75
170 FOR K=Y+1 TO Y+4:Y1=K:FOR X=0 TO 130
 STEP 5:X1=X:GOSUB 200:NEXT X:D=D+.75:NE
XT K,Y
180 Y=Y+1:FOR X=0 TO 130:X1=X:GOSUB 200:
NEXT X
190 GOTO 190
200 XF=(X1-65)/10:YF=(Y1-60)/8
210 IF XF=0 AND YF=0 THEN RETURN
220 Z=M/(XF^2+YF^2):ZT=Z-Y1+180
230 IF ZT>A%(16+X1+D) THEN RETURN
240 PSET(16+X1+D,ZT):A%(16+X1+D)=ZT:RETU
RN
```

ANÁLISE

O programa simula a perspectiva cavaleira do plano distorcido pela função gravidade. As linhas 150 a 190 operam os "loops" de varredura do plano XY ao redor da massa M (situada na origem) e calculam a altura do ponto correspondente.

A sub-rotina das linhas 200 a 240 imprime o ponto nas coordenadas calculadas e coloca sua altura no elemento correspondente do valor A%. Este valor (de 256 elementos) é usado para memorizar a máxima altura impressa em cada uma das 256 colunas da tela e serve para o algorítmo de ocultação dos pontos "invisíveis' (aqueles que estão 'ocultos" por outras imagens que estão mais próximas do observador).

A ocultação dos pontos invisíveis foi obtida pela prioridade de impressão da frente para o fundo da imagem. A linha 230 testa se um ponto deve ou não ser impresso comparando sua altura com o elementos correspondente de A%. Se a altura do ponto for maior, faz-se a impressão. Caso contrário, não.

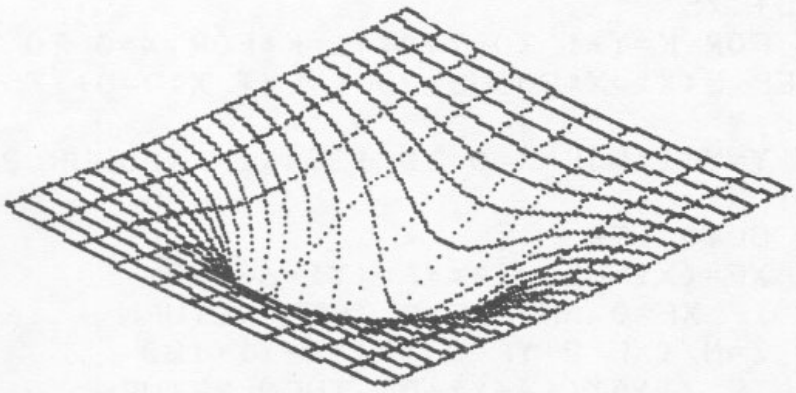
A função matemática propriamente dita está definida na linha 220. Experimente mudar esta função para outros polinômios, como:

$$Z = aX^2 - bY^2$$

Ou:

$$Z = a \times \text{sen}\,(bX^2 + cY^2)$$

Procure os valores a, b e c que dão os melhores resultados. É um pouco difícil, mas vale à pena!



COLEÇÃO DE PROGRAMAS PARA MSX

VOL. I

Editora Aleph

Para certificar-se disso, tecle F1 e verifique se suas alterações coincidem com a figura 1.25. Tecle F5 (RUN) para acionar o programa e veja o efeito de suas alterações.

fig. 1.25

```
10 SCREEN 2
20 DIM A$(8)
30 A$(1)=CHR$(&B11111111)
40 A$(2)=CHR$(&B10000001)
50 A$(3)=CHR$(&B10100101)
60 A$(4)=CHR$(&B10000001)
70 A$(5)=CHR$(&B10100101)
80 A$(6)=CHR$(&B10011001)
90 A$(7)=CHR$(&B10000001)
95 A$(8)=CHR$(&B11111111)
100 FOR I=1 TO 8:B$=B$+A$(I):NEXT I
110 SPRITE$(1)=B$
120 PUT SPRITE 0,(50,85),2,1
130 GOTO 120
```

Se quiser ampliar a figura, breque o programa (CONTROL+STOP), liste-o (F1) e altere a linha 10 para

```
10 SCREEN 2,1
```

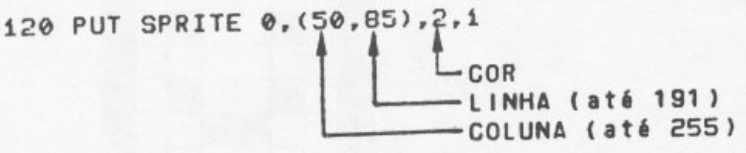
Limpe a tela (SHIFT + HOME/CLS) e rode-o de novo digitando F5.

Agora você pode brincar à vontade, montando a figurinha que quiser e posicionando-a onde quiser.

Para alterar a figurinha, basta alterar a seqüência de 1 e 0 , mantendo-se, porém, dentro de uma tabela 8x8.

A posição e a cor da figurinha são dadas pela linha 120 (fig. 1.26).

fig. 1.26



```
10      Caleidoscopio
20 SCREEN 3:C=84:I=127:D=0:E=D
30 FOR G=E TO C*(1+RND(-TIME)*3)
40 H=INT(36*RND(-TIME)):IF H/4<>H\4 THE
N 40
50 D=INT(D+H*RND(-TIME)):IF D/4<>D\4 TH
EN 50
60 E=INT(E+H*RND(-TIME)):IF E/4<>E\4 TH
EN 60
70 D=D AND D<=C:E=E AND E<=C
80 K=INT(2+13*RND(-TIME)):P$="V15S14M40
00L8N"+STR$(INT(RND(-TIME)*60+9)):PLAY
P$
90 PSET(I+D,C+E),K:PSET(I+D,C-E),K
100 PSET(I-D,C+E),K:PSET(I-D,C-E),K
110 PSET(I+E,C+D),K:PSET(I+E,C-D),K
120 PSET(I-E,C+D),K:PSET(I-E,C-D),K
130 IF INKEY$<>"" THEN CLS
140 NEXT G:GOTO 20
```

ANÁLISE

O funcionamento do CALEIDOSCÓPIO é bastante simples. A linha 20 define a tela a ser usada (no caso, a tela gráfica de baixa resolução) e atribui os valores iniciais às variáveis C, I, D e E.

A linha 30 abre um laço que será repetido um número indefinido de vezes.

As linhas entre 40 e 70 servem para gerar dois números aleatórios e inseri-los nas variáveis D e E. Elas serão usadas para produzir a figura no vídeo.

A linha 80 insere um número entre 2 e 14 na variável K e gera uma nota musical ao acaso.

As linhas de 90 à 120 produzem os pontos da figura na tela.

A linha 130 verifica se você está pressionando alguma tecla e, em caso positivo, limpa a tela.

Finalmente, a linha 140 fecha o laço aberto na linha 30 e, após sua última execução, desvia o programa para a linha 20, reiniciando tudo!

Após ter executado o programa algumas vezes e verificar como ele funciona, experimente alterá-lo deixando-o assim:

```
10 '    Caleidoscopio II
20 SCREEN 2:C=84:I=127:D=0:E=D
```

```
30 FOR G=E TO C*(1+RND(-TIME)*3)
40 H=INT(36*RND(-TIME)):IF H/4<>H\4 THE
N 40
50 D=INT(D+H*RND(-TIME)):IF D/4<>D\4 TH
EN 50
60 E=INT(E+H*RND(-TIME)):IF E/4<>E\4 TH
EN 60
70 D=D AND D<=C:E=E AND E<=C
80 P$="V15S14M4000L8N"+STR$(INT(RND(-TI
ME)*60+9)):PLAY P$
90 PSET(I+D,C+E):PSET(I+D,C-E)
100 PSET(I-D,C+E):PSET(I-D,C-E)
110 PSET(I+E,C+D):PSET(I+E,C-D)
120 PSET(I-E,C+D):PSET(I-E,C-D)
130 IF INKEY$<>"" THEN CLS
140 NEXT G:GOTO 20
```

Ou assim:

```
10       Caleidoscopio III
20 SCREEN 2:C=84:I=127:D=0:E=D
30 FOR G=E TO C*(1+RND(-TIME)*3)
40 H=INT(36*RND(-TIME)):IF H/4<>H\4 THE
N 40
50 D=INT(D+H*RND(-TIME)):IF D/4<>D\4 TH
EN 50
60 E=INT(E+H*RND(-TIME)):IF E/4<>E\4 TH
EN 60
70 D=D AND D<=C:E=E AND E<=C
80 P$="V15S14M4000L8N"+STR$(INT(RND(-TI
ME)*60+9)):PLAY P$
90 LINE(I,C)-(I+D,C+E):LINE(I,C)-(I+D,C
-E)
100 LINE(I,C)-(I-D,C+E):LINE(I,C)-(I-D,
C-E)
110 LINE(I,C)-(I+E,C+D):LINE(I,C)-(I+E,
C-D)
120 LINE(I,C)-(I-E,C+D):LINE(I,C)-(I-E,
C-D)
130 IF INKEY$<>"" THEN CLS
140 NEXT G:GOTO 20
```

Ou ainda, assim:

```
10 '    Caleidoscopio IV
20 SCREEN 2:C=84:I=127:D=0:E=D
30 FOR G=E TO C*(1+RND(-TIME)*3)
40 H=INT(36*RND(-TIME)):IF H/4<>H\4 THE
N 40
50 D=INT(D+H*RND(-TIME)):IF D/4<>D\4 TH
EN 50
60 E=INT(E+H*RND(-TIME)):IF E/4<>E\4 TH
EN 60
70 D=D AND D<=C:E=E AND E<=C
80 P$="V15S14M4000L8N"+STR$(INT(RND(-TI
ME)*60+9)):PLAY P$
90 LINE(I+D,C+E)-(I+D,C-E)
91 LINE(I+D,C-E)-(I-D,C-E)
92 LINE(I-D,C-E)-(I-D,C+E)
93 LINE(I-D,C+E)-(I+D,C+E)
94 LINE(I+E,C+D)-(I+E,C-D)
95 LINE(I+E,C-D)-(I-E,C-D)
96 LINE(I-E,C-D)-(I-E,C+D)
97 LINE(I-E,C+D)-(I+E,C+D)
130 IF INKEY$<>"" THEN CLS
140 NEXT G:GOTO 20
```

Agora o efeito não parece mais com um caleidoscópio porém continua a ser belo.

Não há nenhum segredo em suas utilizações. Basta digitá-los e executá-los!

Se a execução demorar muito, faça alguma outra coisa enquanto espera. Por exemplo, vá lendo a análise e tentando entender o funcionamento dos programas.

DIGITAÇÃO

Estes programas possuem muitas instruções inseridas numa mesma linha e separadas por dois pontos (:). Tome muito cuidado para não pular algumas delas, pois isso pode alterar totalmente os efeitos produzidos.

Os programas são curtos. Em cinco minutos você digita qualquer um deles. Vale a pena tomar um pouco mais de cuidado durante a primeira digitação para não perder tempo procurando eventuais erros depois.

TWO-LINER 1 - CIRCLINE

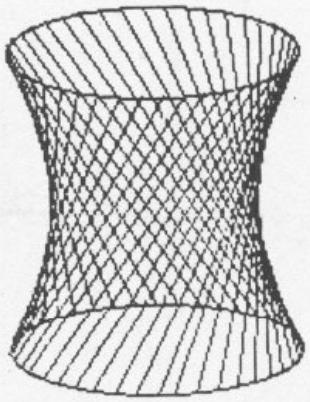
```
10 SCREEN 2
20 CC=128:LC=80:R=60:FORT=0TO359STEP5:G
G=2+14*RND(-TIME):X=T*3.141592#/180:C=C
C+R*COS(X):L=LC+R*SIN(X):LINE(CC,LC)-(C
,L),GG,B:NEXT:GOTO20
```

TWO-LINER 2 - CORES

```
10 SCREEN2
20 FORF=0TO125STEP10:G=80-F*80/125:G1=2
+14*RND(-TIME):G2=2+14*RND(-TIME):G3=2+
14*RND(-TIME):G4=2+14*RND(-TIME):LINE(F
,80)-(125,G),G1,BF:LINE(125,G)-(250-F,8
0),G2,BF:LINE(F,80)-(125,160-G),G3,BF:L
INE(125,160-G)-(250-F,80),G4,BF:NEXT:GO
TO20
```

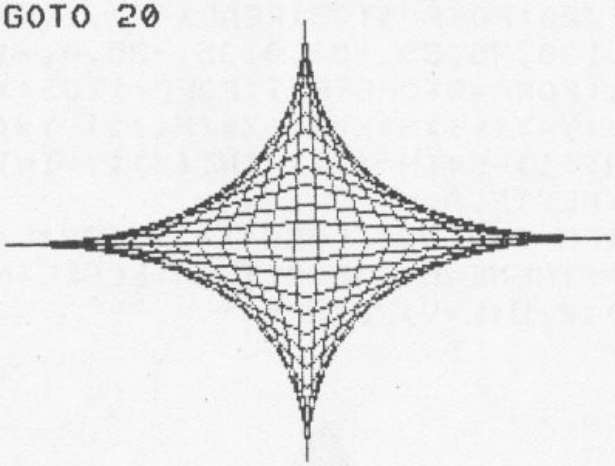
TWO-LINER 3 - CESTO

```
10 SCREEN2:P=3.1415926#:CIRCLE(128,30),
60,,,,20/60:CIRCLE(128,160),60,,,,20/60
:FORD=0TO2*PSTEP.15:E=D+P/2:X=128+60*CO
S(D):Y=30+20*SIN(D):S=128+60*COS(E):T=1
60+20*SIN(E):LINE(X,Y)-(S,T):NEXTD
20 GOTO 20
```



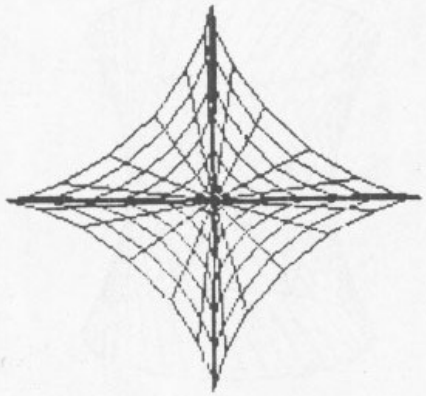
TWO-LINER 4 - ASTRÓIDE 1 (Rede)

```
10 SCREEN2:FORF=0TO125STEP10:G=80-F*80/
125:LINE(F,80)-(125,G):LINE(125,G)-(250
-F,80):LINE(F,80)-(125,160-G):LINE(125,
160-G)-(250-F,80):NEXT
20 GOTO 20
```



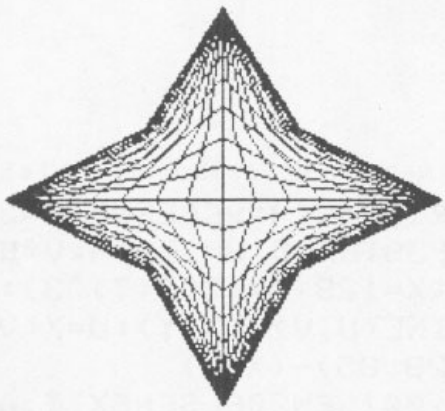
TWO-LINER 5 - ASTRÓIDE 2 (Teia)

```
10 COLOR1,14,14:SCREEN2:P=3.1415926#:FO
RA=10TO85STEP15:U=128+A:V=85:FORT=0TO2*
PSTEP.2:X=128+A*(COS(T)^3):Y=85+A*(SIN(
T)^3):LINE(U,V)-(X,Y):U=X:V=Y:IFA=85THE
NLINE(128,85)-(X,Y)
20 IFA=100THEN20ELSENEXTT,A:GOTO20
```



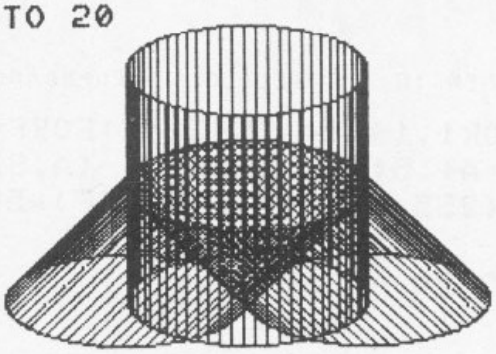
TWO-LINER 6 - ASTRÓIDE 3 (Estrela)

```
10 COLOR10,1:SCREEN2:P=3.1415926#:C=2*P
:I=P/20:FORF=1TO5:READX(F),Y(F):NEXT:DA
TA0,100,35,35,100,0,35,-35,0,-100:FORH=
0TO1:FORA=0TOPSTEPI:FORF=1TO5:X=X(F)*CO
S(A):Y=Y(F):S=X:X=-X*(H<>1)-Y*(H=1):Y=-
Y*(H<>1)-S*(H=1):X=INT(X):Y=INT(Y):GOSU
B20:NEXTF,A,H
20 IFH=2THEN20ELSEU=X*.9+128:V=Y*.9+85:
IFF=1THENK=U:L=V:RETURN:ELSELINE(K,L)-(
U,V):K=U:L=V:RETURN
```



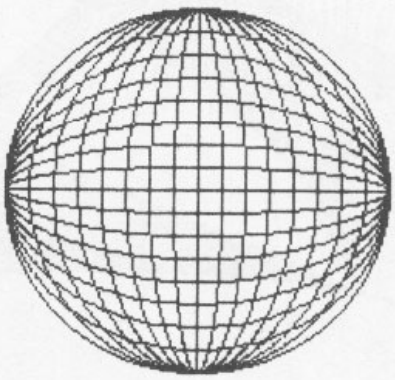
TWO-LINER 7 - DRAWLIPSE

```
10 COLOR1,8,8:SCREEN2:FORA=ØTO6.28STEP
.1:PSET(128+50*COS(A),85+20*SIN(A)):DRA
W"nu50nf50nd50ng50":NEXTA:CIRCLE(128,34
),50,1,,,,.4:CIRCLE(128,135),50,1,,,,.4:C
IRCLE(128,85),50,1,,,,.4:CIRCLE(77,135),
50,1,,,,.4:CIRCLE(178,135),50,1,,,,.4
20 GOTO 20
```



TWO-LINER 8 - ESFERA

```
10 COLOR1,7,7:SCREEN2:FORB=80TO1STEP-10
:CIRCLE(128,80),80,1,,,80/B:CIRCLE(128,
80),80,1,,,B/80:NEXTB:LINE(128,160)-(12
8,0):LINE(48,80)-(208,80)
20 GOTO 20
```



TWO-LINER 9 - MONTES

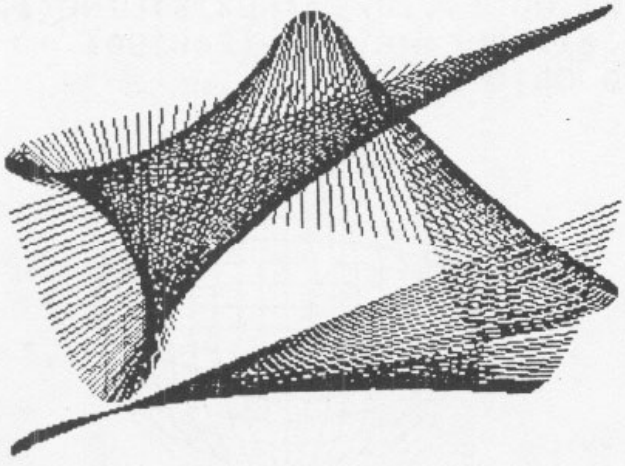
```
10 COLOR4,7,7:SCREEN2:FORX=0TO191:X1=X1
+255/191:P=X1*ATN(1)/45*360/255*3:Y=COS
(P)*130+128:LINE(X1,0)-(Y,X):NEXTX
20 GOTO 20
```

TWO-LINER 10 - TECIDO 1 (Enrugado)

```
10 COLOR1,14,14:SCREEN2:FORF=0TO25.2STE
P.05:A=A+.5:LINE(255,0)-(A,SIN(F)*50+90
):LINE(255,0)-(A-.5,SIN(F)*50+88),14:NE
XTF
20 GOTO 20
```

TWO-LINER 11 - TECIDO 2 (Torcido)

```
10 IFA=192THEN10ELSECOLOR4,7,5:SCREEN2:
FORA=0TO191:B=A*ATN(1)/90*360/255*4:C=C
+255/191:D=C*ATN(1)/90*360/255*3
20 E=SIN(B)*85+85:F=COS(D)*127+127:LINE
(C,E)-(F,A):NEXTA:GOTO 10
```

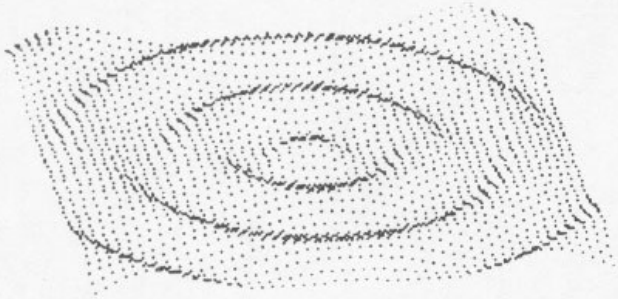


TWO-LINER 13 - COGUMELO 1

```
10 SCREEN2
20 FORZ=0TO10STEP.1:FORX=0TO10STEP.1:Y=
-(COS(3*SQR((X-5)*(X-5)+(Z-5)*(Z-5))))*
5+30:PSET(X*24+Z,191-Y-Z*10):NEXTX,Z:GO
TO20
```

TWO-LINER 14 - COGUMELO 2

```
10 SCREEN2:FORZ=0TO10STEP.2:FORX=0TO10S
TEP.2:Y=-10*COS(3*SQR((X-5)*(X-5)+(Z-5)
*(Z-5)))/2+50:PSET(X*20+20+Z*3,Y+Z*10):
NEXTX,Z
70 GOTO 70
```

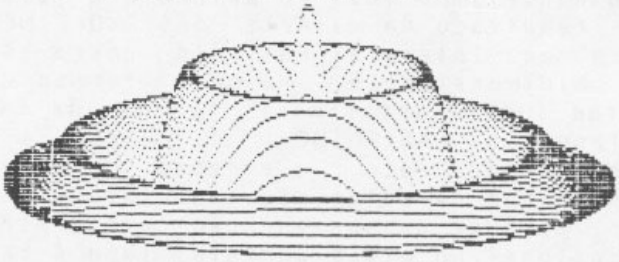


TWO-LINER 15 - COGUMELO 3

```
10 COLOR1,3,3:SCREEN2:FORB=0TO127STEP2:
X4=B*B:M=-128:A=SQR(16384-X4):FORI=-ATO
ASTEP5:R=SQR(X4+I*I)/128:F=COS(9*R)*(1-
R)*2:Y=I/5+F*32:IFY<=MTHEN20ELSEM=Y:Y=1
28+Y:X=128+B:PSET(X,191-Y):X=128-B:PSET
(X,191-Y)
20 IFP=1THEN20ELSENEXTI,B:P=1:GOTO20
```

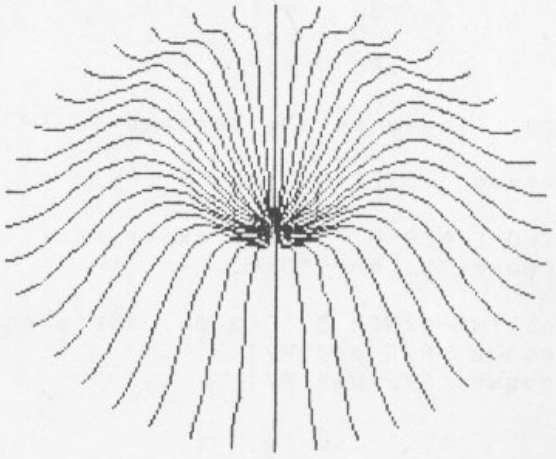
TWO-LINER 16 - COGUMELO 4

```
10 COLOR1,5,5:SCREEN2:FORB=0TO127:X4=B*
B:M=-128:A=SQR(16384-X4):FORI=-ATOASTEP
3:R=SQR(X4+I*I)/128:F=COS(16*R)*(1-R)*2
:Y=I/5+F*32:IFY(=MTHEN20ELSEM=Y:Y=128+Y
:X=128+B:PSET(X,191-Y):X=128-B:PSET(X,1
91-Y)
20 IFP=1THEN20ELSENEXTI,B:P=1:GOTO20
```



TWO-LINER 17 - ANÊMONA

```
10 COLOR1,3,3:SCREEN2:P=3.1415926#:Q=2*
P:R=P/20:S=P/10:T=60:U=150/Q:FORB=0TOQS
TEPR:C=T*P/180:C=(B>P)*C-(B(=P)*C:FORH=
0TOQSTEPS:Y=COS(H)*42:X=COS(B)*H*U:Z=-(
SIN(B)=1)*H*U-(SIN(B)()1)*TAN(B)*X:Z=AB
S(Z):Z=-1E-03*(Z=0)-Z*(Z()0):J=SQR(Z*Z+
Y*Y):A=ATN(Y/Z)+C
20 IFF=1THEN20ELSEY=J*SIN(A):X=INT(X):Y
=INT(Y):XP=X*.7+128:YP=Y*.7+85:XO=-XO*(
H()0)-XP*(H=0):YO=-YO*(H()0)-XP*(H=0):L
INE(XO,YO)-(XP,YP):XO=XP:YO=YP:NEXTH,B:
F=1:GOTO20
```

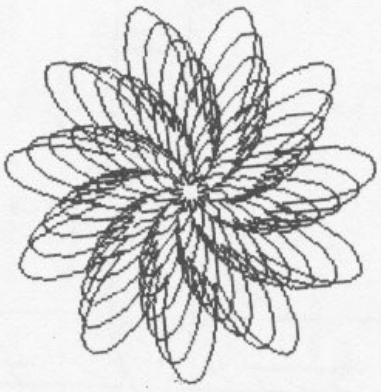


Agora é a sua vez de propor (e realizar!) alterações. Comece pelo primeiro programa e siga até o último.

Depois, terá chegado a hora de você fazer seus próprios TWO-LINERS. Inicialmente, faça o programa em várias linhas. A seguir, compacte-o em apenas duas. Isso, ao contrário do que parece, não é muito simples! Para certificar-se disso, digite e execute algumas vezes o THREE-LINER Pétalas. Depois, tente transformá-lo num TWO-LINER! Se você coinseguir, avise-nos! Isso é extremamente difícilllr

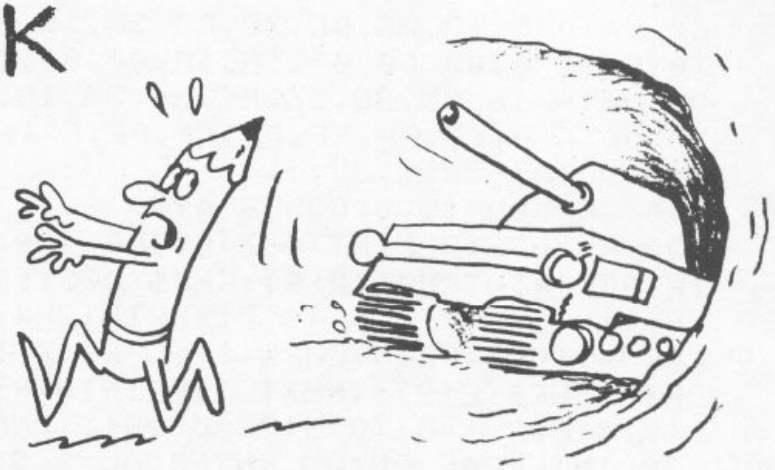
THREE-LINER 1 - PÉTALAS

```
10 INPUT"Quantas petalas";P:INPUT"Com r
otacao (s/n)";A$:PI=3.141592#:Q=2*PI:CO
LOR1,8,8:SCREEN2:N1=5:N2=4:FORV=1TO10:I
=V*PI/30*(A$="s"):M1=N1*V:M2=N2*V:FORF=
0TOQSTEPQ/180:A=F-I:S=ABS(SIN(F*P/2)):R
=S*M1+M2:X=COS(A)*R:Y=SIN(A)*R:GOSUB30:
X=INT(X):Y=INT(Y)
20 IFIP=1THEN20ELSENEXTF:GOSUB30:NEXTV:
IP=1:GOTO 20
30 XP=X*.9+128:YP=191-(Y*.9+85):IFF=0TH
ENX1=X:Y1=Y:XO=XP:YO=YP:RETURN20:ELSELI
NE(XO,YO)-(XP,YP):XO=XP:YO=YP:RETURN
```



Este programa pede a introdução de duas informações: o número de pétalas e se elas devem ser rotacio-

TANK



INSTRUÇÕES

Neste jogo sua missão é roubar um diamante que está guardado no fundo de um subterrâneo protegido por quatro robôs guardiães e em seguida trazê-lo de volta à superfície. Você dispõe de um tanque de guerra que pode abrir caminho a tiros e repelir os robôs que tentarão cercá-lo. Para pegar o diamante, basta passar com o tanque por cima dele.

O tanque é controlado pelas teclas de cursor e barra de espaço. As teclas ◁ e ▷ giram o tanque, a tecla △ o move para a frente e a barra de espaços é usada para atirar.

A saída fica no canto superior esquerdo da tela, sobre o lugar onde o tanque é colocado inicialmente. Não tente sair sem ter pego o diamante (você perceberá tardiamente que isto é desastroso e extremamente desaconselhável). Se mesmo assim você quiser experimentar, ao menos grave o programa antes !!!

DIGITAÇÃO

O programa não oferece dificuldade de digitação, mas é importante transcrever as linhas com bastante atenção para não haver surpresas na hora de rodar o programa.

```
1 '     TANK ATTACK
10 DATA 24,10,10,D6,D6,FE,FE,FE,C6
20 DATA 25,FC,FC,70,7F,70,FC,FC,00
30 DATA 26,C6,FE,FE,FE,D6,D6,10,10
40 DATA 27,3F,3F,0E,FE,0E,3F,3F,00
50 DATA 4,18,3C,3D,3E,3D,3C,18,3C
```

```
60 DATA 5,18,3C,BC,7C,BC,3C,18,3C
70 DATA 6,00,00,00,18,18,00,00,00
80 DATA 16,00,3C,56,FF,56,34,18,00
90 DATA 8,FF,FF,FF,FF,FF,FF,FF,FF
100 DATA -32,1,32,-1
110 SCREEN 1,,0:GOSUB 490
120 KEYOFF:DEFINT A-Z:GOSUB 450:CLS:WID
TH 30:PRINTCHR$(219);CHR$(203);STRING$(
28,219);:FOR I=1 TO 21:PRINTCHR$(219);:
FORJ=1TO28:PRINTCHR$(1);CHR$(72);:NEXT:
PRINTCHR$(219);:NEXT I:PRINTSTRING$(30,
219);:FOR I=0 TO 3:READ DS(I):NEXT
130 INTERVAL ON:ON INTERVAL=3 GOSUB 340
:DEF FNPO(X,Y)=6144+X+32*Y:VPOKE 6831,1
6
140 A=6178:D=2:F=0:P=0:VPOKE A,D+24:STR
IG(0)OFF:ON STRIG GOSUB 330:FOR I=1 TO
4:X(I)=15+I:Y(I)=15:NEXT I
150 STRIG(0)OFF:C=STICK(0):IF C=3THEND=
D+1ELSEIFC=7THEND=D-1ELSE170
160 OUT170,154:OUT170,26
170 IF D=-1THEND=3ELSEIFD=4THEND=0
180 VPOKE A,D+24:STRIG(0)ON
190 X=(A-6144)MOD32:Y=(A-6144)\32:FOR I
=1 TO 4:Z=8:IF I<4THENZ=32
200 VPOKEFNPO(X(I),Y(I)),Z:Z=4:DX=0:DY=
0:IFRND(1)>.5THENDX=SGN(X-X(I))ELSEDY=S
GN(Y-Y(I))
210 IF I<4THEN 250
220 IFVPEEK(FNPO(X(I)+DX,Y(I)))<>32THEN
IFVPEEK(FNPO(X(I)-DX,Y(I)))=32THENDX=-D
XELSEDX=0
230 IFVPEEK(FNPO(X(I),Y(I)+DY))<>32THEN
IFVPEEK(FNPO(X(I),Y(I)-DY))=32THENDY=-D
YELSEDY=0
240 U=FNPO(X(I),Y(I)):IF VPEEK(U+1)+VPE
EK(U-1)+VPEEK(U+32)+VPEEK(U-32)=32THENG
OSUB400
250 X(I)=X(I)+DX:Y(I)=Y(I)+DY:IFX(I)>XT
HENZ=5
```

```
260 IFVPEEK(FNPO(X(I),Y(I)))=6THENX(I)=
20:Y(I)=15
270 K=VPEEK(FNPO(X(I),Y(I))):IFK>23ANDK
<28THENT=0:STRIG(0)OFF:GOTO410ELSEVPOKE
FNPO(X(I),Y(I)),Z:NEXT I
280 K=VPEEK(DS(D)+A):IFC<>1THEN320
290 IFK<>8ANDK<>219THEN VPOKE A,32:A=A+
DS(D):VPOKE A,D+24:OUT 170,154:OUT 170,
26
300 IFK=16THENF=1:FORI=1TO10:BEEP:NEXTI
:ELSEIFK=40RK=5THENT=0:STRIG(0)OFF:GOTO
410
310 IFK=203THENIFF=1THEN420ELSECLS:PRIN
T"VOCE NAO DEVIA TENTAR FUGIR...":NEW
320 GOTO 150
330 IF T>0 THEN RETURN ELSE T=A:DT=D:BE
EP:INTERVAL ON:RETURN
340 IF T=0 THEN RETURN
350 IF VPEEK(T)=6 THEN VPOKE T,32
360 T=T+DS(DT):IF VPEEK(T)=32 THEN VPOK
E T,6:RETURN
370 IF VPEEK(T)=8 THEN VPOKE T,32
380 IFVPEEK(T)=40RVPEEK(T)=5THENGOSUB47
0
390 INTERVAL OFF:T=0:RETURN
400 U=FNPO(2,Y(I)):FOR J=U TO U+27:VPOK
E J,32:NEXT J:RETURN
410 CLS:LOCATE 0,10:PRINT"VOCE FOI CAPT
URADO":GOTO430
420 CLS:LOCATE 0,10:PRINT"VOCE CONSEGUI
U"
430 PRINT"JOGA DE NOVO ?";
440 A$=INPUT$(1):IFA$="S"THENRUNELSEIFA
$="N"THENENDELSE440
450 CLS:FOR I=0 TO 8:READ Z:FOR J=0 TO
7:READ B$:VPOKE J+8*Z,VAL("&H"+B$):NEXT
 J:VPOKE 6150+2*I,Z:NEXT I
460 I$=INPUT$(1):RETURN
470 FOR I=1TO4:IFFNPO(X(I),Y(I))<>TTHEN
NEXTI:RETURN
```

```
480 VPOKE T,32:X(I)=20:Y(I)=15:FORJ=1TO
5:OUT170,154:OUT170,26:NEXTJ:RETURN
490 VPOKE8192,135:VPOKE8193,96
500 VPOKE8194,246:VPOKE8195,39:FORI=819
6TO8210:VPOKEI,7:NEXTI:VPOKE8219,176
510 RETURN
```

ANÁLISE

O programa inicia chamando uma sub-rotina (linhas 450-510) que lê as instruções DATA do início do programa e faz a redefinição dos caracteres (tanque, robôs, diamante, etc).

Em seguida, o programa imprime o cenário (linha 120) e inicia as variáveis (linhas 120-140). A ação central do jogo fica entre as linhas 330 e 390. A linha 400 serve para abrir uma passagem para o robô obturado (o que não escava túneis) e as linhas 410 à 440 encerram o jogo.

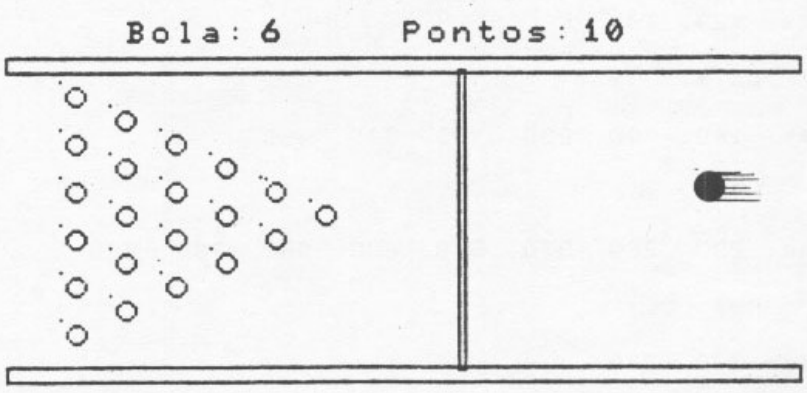
BOLICHE

INSTRUÇÕES

Este jogo é extremamente simples, a despeito de seu programa ser um tanto complexo.

Ao ser executado, ele gera na tela a imagem de uma pista de boliche vista de cima. Existem 21 garrafas que deverão ser derrubadas com 6 bolas.

fig. 12.1



A bola surge numa posição à direita da pista e vai se deslocando para a esquerda. Você pode movê-la para cima ou para baixo com as setas △e▽, porém apenas enquanto ela não chegou à faixa demarcatória central. Após esse limite, a bola segue uma linha reta.

Ao inserirmos este programa neste livro, não pensamos em suas qualidades como jogo, mas sim no exemplo que ele representa quanto à utilização da instrução ON SPRITE.

* linhas: 540, 580, 590, 630, 640, 650, 690, 700, 710, 720, 750, 760, 770, 780, 790

IF....THEN....PUT SPRITE....

* linhas: 10, 30, 140, 290, 480, 520, 560, 610, 670, 740, Linhas REM

As linhas não inseridas em nenhum desses grupos são as seguintes: 10, 20, 150, 160, 170, 280, 310, 380, 390, 400, 440, 490, 550.

Após digitar completamente o programa, qualquer que seja a sequência escolhida, confira-o, ao menos uma vez, linha por linha.

```
10 ' B O L I C H E
20 SCREEN 2,0,0
30 '########################################
40 FOR F=1 TO 8:READ A$
50 S$=S$+CHR$(VAL("&H"+A$)):NEXT F
60 SPRITE$(1)=S$:S$=""
70 FOR F=1 TO 8:READ A$
80 S$=S$+CHR$(VAL("&H"+A$)):NEXT F
90 SPRITE$(2)=S$:S$=""
100 FOR F=1 TO 8:S$=S$+CHR$(0):NEXT F
110 SPRITE$(3)=S$
120 DATA 3C,66,FB,FD,FF,FF,7E,3C
130 DATA 38,44,BA,BA,BA,44,38,00
140 '########################################
150 MAXFILES=1:PT=0:COLOR 15,6,6:CLS
160 OPEN"GRP:" FOR OUTPUT AS #1
170 PRESET(60,2):PRINT #1,"     BOLICHE

                Bola:       Pontos:"
180 LINE(0,30)-(255,35),1,BF
190 LINE(0,134)-(255,139),1,BF
200 LINE(145,34)-(147,135),1,B:A=1
210 FOR K=0 TO 5:X=K*16+20
220 FOR Y=40+8*K TO 80 STEP 16
230 PUT SPRITE A,(X,Y),15,2:A=A+1
240 CIRCLE(X+2,Y+2),3,1:PAINT(X+2,Y+2),
    1
250 IF Y=80 THEN 280
260 PUT SPRITE A,(X,160-Y),15,2:A=A+1
```

```
270 CIRCLE(X+2,162-Y),3,1:PAINT(X+2,162
-Y),1
280 NEXT Y,K
290 '########################################
300 FOR S=1 TO 6
310 Y=36+INT(RND(-TIME)*78)
320 LINE(78,19)-(90,28),6,BF
330 PRESET(70,19):PRINT#1,S
340 LINE(182,19)-(220,28),6,BF
350 PRESET(175,19):PRINT#1,PT
360 FOR X=255 TO 0 STEP-1
370 PUT SPRITE 0,(X,Y),1,1
380 SPRITE ON
390 ON SPRITE GOSUB 490
400 T=STICK(0)
410 IF X<145 THEN 440
420 IF T=5 THEN Y=Y+1:IF Y>125 THEN Y=1
25
430 IF T=1 THEN Y=Y-1:IF Y<35 THEN Y=35
440 NEXT X,S
450 LINE(182,19)-(220,28),6,BF
460 PRESET(175,19):PRINT#1,PT
470 IF STRIG(0)=-1 THEN RUN ELSE 470
480 '########################################
490 SPRITE OFF:BEEP:PT=PT+1
500 IF X<100 THEN 530
510 PUT SPRITE 21,(X,Y),0,3:RETURN
520 '########################################
530 IF X<84 THEN 570
540 IF Y<80 THEN PUT SPRITE 19,(X,Y),0,
3 ELSE PUT SPRITE 20,(X,Y),0,3
550 RETURN
560 '########################################
570 IF X<68 THEN 620
580 IF Y<72 THEN PUT SPRITE 16,(X,Y),0,
3:RETURN
590 IF Y<88 THEN PUT SPRITE 18,(X,Y),0,
3:RETURN
600 PUT SPRITE 17,(X,Y),0,3:RETURN
610 '########################################
```

```
620 IF X<52 THEN 680
630 IF Y<64 THEN PUT SPRITE 12,(X,Y),0,
3:RETURN
640 IF Y<80 THEN PUT SPRITE 14,(X,Y),0,
3:RETURN
650 IF Y<96 THEN PUT SPRITE 15,(X,Y),0,
3:RETURN
660 PUT SPRITE 13,(X,Y),0,3:RETURN
670 '####################################
680 IF X<36 THEN 750
690 IF Y<56 THEN PUT SPRITE 7,(X,Y),0,3
:RETURN
700 IF Y<72 THEN PUT SPRITE 9,(X,Y),0,3
:RETURN
710 IF Y<88 THEN PUT SPRITE 11,(X,Y),0,
3:RETURN
720 IF Y<104 THEN PUT SPRITE 10,(X,Y),0
,3:RETURN
730 PUT SPRITE 8,(X,Y),0,3:RETURN
740 '####################################
750 IF Y<48 THEN PUT SPRITE 1,(X,Y),0,3
:RETURN
760 IF Y<64 THEN PUT SPRITE 3,(X,Y),0,3
:RETURN
770 IF Y<80 THEN PUT SPRITE 5,(X,Y),0,3
:RETURN
780 IF Y<96 THEN PUT SPRITE 6,(X,Y),0,3
:RETURN
790 IF Y<112 THEN PUT SPRITE 4,(X,Y),0,
3:RETURN
800 PUT SPRITE 2,(X,Y),0,3:RETURN
```

ANÁLISE

Para facilitar a análise do programa, vamos dividi-lo em 4 blocos principais:

1) Bloco de definição de SPRITEs
   linha inicial- 40
   linha final- 130

Chegamos, por fim, à sub-rotina que constitui o bloco de verificação das garrafas.

Quando a bola atinge uma garrafa, o programa deve apagá-la da tela. O problema está em determinar em qual camada está a garrafa atingida, pois isso é essencial para apagá-la.

Existem, inicialmente, 21 garrafas no vídeo, cada uma em uma camada (de 1 a 21).

A rotina a partir da linha 490 verifica a garrafa atingida através da posição da bola na tela.

Vejamos como isso é feito.

Quando a bola atinge alguma garrafa, ocorre uma sobreposição de SPRITEs e a linha 390 desvia o programa para a linha 490. Essa linha "desliga" o desvio produzido pela linha 390, gera um "beep" e incrementa a pontuação do jogador. Se a instrução SPRITE OFF fosse eliminada da linha 490, o desvio produzido pela linha 390 não seria desligado e o programa ficaria indefinidamente "parado" assim que alguma garrafa fosse atingida pela bola. Experimente eliminar a instrução SPRITE OFF para compreender melhor porque ela é essencial.

Logo depois, na linha 500, verifica-se se a garrafa mais à direita da tela foi atingida. Essa garrafa está na camada 21 e suas coordenadas são X=100 e Y=80. Se a coordenada X da bola for maior ou igual a 100, certamente a garrafa atingida foi a da camada 21 (veja a fig. 12.2).

Caso a coordenada X da bola seja menor que 100, alguma outra garrafa foi atingida e o programa vai para a linha 530.

Se a coordenada X da bola for maior ou igual a 84, significa que a garrafa atingida foi a da camada 19 ou da camada 20. Nesse caso, a decisão é feita com base na coordenada Y da bola. Se Y é menor que 80, a garrafa atingida é a da camada 19, e se Y for maior ou igual a 80, a garrafa é a da camada 20.

Se a coordenada X da bola for menor que 84, o programa é desviado para a linha 570, pois a garrafa atingida foi alguma outra.

Assim, o programa prossegue na sub-rotina, averiguando garrafa por garrafa, até achar a que foi atingida. Então retorna à rotina principal.

Essa versão do BOLICHE necessita de todas essas linhas porque usa SPRITEs. Um efeito mais modesto é obtido com o programa a seguir que utiliza a tela de baixa resolução.

```
10 ' B O L I C H E  II
20 COLOR 15,6,6:SCREEN3
30 OPEN"GRP:"FOROUTPUTAS#1
40 PRESET(24,0):PRINT#1,"BOLICHE":PT=0
50 LINE(0,30)-(255,35),1,BF
```

```
60 LINE(0,128)-(255,133),1,BF
70 FOR K=0 TO 5:X=K*16+20
80 FOR Y=40+8*KTO80STEP16:PSET(X,Y)
90 IF Y()80 THEN PSET(X,160-Y)
100 NEXT Y,K
110 FOR S=1 TO 11:A%=32000!*RND(-TIME)
120 LINE(10,150)-(90,191),6,BF
130 PRESET(10,150):PRINT#1,S
140 LINE(140,150)-(250,191),6,BF
150 PRESET(140,150):PRINT#1,PT
160 Y=36+4*INT(RND(A%)*20)
170 FOR X=255 TO 0 STEP-4
180 IF POINT(X,Y)=15 THEN PT=PT+1
190 FORF=1TO5:NEXTF
200 PSET(X,Y),1:PRESET(X+4,J)
210 T=STICK(0):J=Y
220 IFT=5THENY=Y+4:IFY)124THENY=124
230 IFT=1THENY=Y-4:IFY(36THENY=36
240 NEXT X,S
250 LINE(140,150)-(250,191),6,BF
260 PRESET(140,150):PRINT#1,PT
270 IF STRIG(0)=0 THEN 270 ELSE RUN
```

Muitas alterações podem ser feitas no programa que usa SPRITEs a fim de melhorá-lo. Uma delas pode ser a substituição do "beep" da linha 490 pelo barulho de choque característico do boliche. Outra alteração seria a opção de mais de um jogador, com a indicação do recorde e dos nomes. Espaço na tela há de sobra para isso!

Com um pouco de imaginação, você mesmo pode implementar seu programa!

```
10 '       CRAB-CANON
20 PLAY"v10t120","v9t120"
30 PLAY"o312ce-gf","o314ce-go4c18o3bo4c
de-fe-dc"
40 PLAY"o2112br4o312gf#","o418do3go3dfe
-dco3babo4ce-"
50 PLAY"o312fee-d","o418dco3baga-b-o4d-
co3b-a-gfgfga-a-"
60 PLAY"o314dd-co2bgo3ce-","o318b-gfe-d
c-fgfe-db-"
70 PLAY"o312e-dce-","o318gfe-o4co3bagfe
-de-go4co3gfg"
80 PLAY"o318gfgo4co3ge-de-fgabo4co3e-fg
","o312e-cde-"
90 PLAY"o318a-de-fgfe-de-fga-b","o314fc
o2gbo3cd-"
100 PLAY"o318b-a-gfga-b-o4cd-o3b-a-gabo
4d","o312e-eff#"
110 PLAY"o418e-co3babo4cde-fdo3go4dcde-
f","o314ggr4o2bbo3aa"
120 PLAY"o418e-dco3bo4co3ge-c","o312ge-
c"
```

ANÁLISE.

Todas as linhas do programa têm a mesma função: gerar sons musicais através de dois canais independentes. A sintaxe usada da instrução PLAY em todas as linhas foi:

```
PLAY"sub-comandos musicais da primeira voz",
   "sub-comandos musicais da segunda voz"
```

Os sub-comandos à esquerda da vírgula são executados através do canal 1 e, independentemente, os sub-comandos à direita da vírgula são executados através do canal 2.

TWO LINERS
(SONORO)

INSTRUÇÕES

Os seis programas aqui apresentados geram efeitos sonoros usando o Gerador de Som Programável (PSG) do MSX.

Você poderá usá-los como sub-rotinas para implementar jogos ou sinalizar programas utilitários.

Digite-os, um de cada vez, e comande RUN. Para ouví-los com maior qualidade, você pode também ligar o Expert a um amplificador.

DIGITAÇÃO

Não há dificuldade em digitar nenhum dos seis programas. Todos são bastante curtos e simples.

ATERRISSAGEM

```
10 SOUND 7,56:SOUND 8,15:SOUND 9,0:SOUN
D 10,0:FOR L=0 TO 511:FOR M=L TO L+50 S
TEP 8:SOUND 0,M MOD 256:SOUND 1,M\256:N
EXT M,L
20 SOUND 7,7:SOUND 8,16:SOUND 12,32:SOU
ND 6,0:SOUND 13,15:FOR I=1 TO 500:NEXT
I:SOUND 13,0:END
```

CAMPAINHA

```
10 DATA 100,125,112,170,170,112,100,125
20 RESTORE:SOUND 7,56:SOUND 8,16:SOUND
```

```
1,0:SOUND 12,36:FOR I=1 TO 8:READ A:SOU
ND 0,A:SOUND 13,0:FOR L=1 TO 400:NEXT L
:IF I=4 THEN FOR L=1 TO 400:NEXT L:NEXT
 I ELSE NEXT I:END
```

ALARME 1

```
10 SOUND 7,56:SOUND 8,15:SOUND 1,0
20 SOUND 0,50:FOR I=1 TO 200:NEXT I:SOU
ND 0,100:FOR I=1 TO 200:NEXT I:GOTO 20
```

ALARME 2

```
10 SOUND 7,56:SOUND 8,15:SOUND 1,0
20 FOR L=230 TO 100 STEP -.5:SOUND 0,L:
NEXT L:SOUND 0,0:FOR I=1 TO 150:NEXT I:
GOTO 20
```

SIRENE

```
10 SOUND 7,56:SOUND 8,15:SOUND 1,0:SOUN
D 9,15:SOUND 3,0
20 SOUND 0,190:SOUND 2,192:FOR I=1 TO 4
00:NEXT I:SOUND 0,250:SOUND 2,253:FOR I
=1 TO 400:NEXT I:GOTO 20
```

DECOLAGEM

```
10 SOUND 7,28:SOUND 1,0:SOUND 3,0:SOUND
 6,0:SOUND 0,100:SOUND 8,13:SOUND 9,0:S
OUND 10,15:SOUND 12,255:FOR L=1 TO 2000
:NEXT L
20 FOR L=100 TO 30 STEP -.04:SOUND 0,L:
NEXT L:SOUND 8,0:FOR L=2 TO 31 STEP .01
:SOUND 6,L:NEXT L:SOUND 10,16:SOUND 13,
0
```

ANÁLISE

A programação do Gerador de Som Programável do MSX é feita de forma direta através do BASIC. A instrução usada é o SOUND, e para entender melhor como ela funciona, consulte o livro LINGUAGEM BASIC MSX , a partir da página 144.

Este livro foi impresso pela
artes gráficas guaru s/a.
Rod. Presidente Dutra, km 214
Fone: 912-1388 - Guarulhos