BIBLIOTECA BÁSICA

INFORMÁTICA

DESENHO DE JOGOS 17

SECULO FUTURO

BIBLIOTECA BÁSICA

INFORMÁTICA

DESENHO DE JOGOS 17

SÉCULO FUTURO

Diretor editor:
M. A. Nieto

Coordenação e supervisão técnica:
Engº Sergio Rocha Paggioli

Tradução:
Ideli Novo

Projetos especiais:
Rainer K.E. Ladewig

Editor Gráfico:
Auro Pereira da Silva

Revisão:
Susana M. Amaral Couto

Produção Gráfica:
Luiz Carlos Siqueira Lago

Arte-Final:
Rubens Tadeu Benedito (coordenador)
Luiz Antonio de Andrade
Vadinho de Oliveira

Fotocomposição, fotolito:
Omnicolor Gráfica e Propaganda Ltda - Rua Dr. Virgílio de Carvalho Pinto, 619
Pinheiros - CEP 05415 - São Paulo.

Impressão:
Editora Antártica S.A. - Av. Ramon Freire, 6920 (Pajaritos) - Santiago - Chile

ÍNDICE

PREFÁCIO

05 Prefácio

CAPÍTULO I

07 A aventura dos jogos de aventura

CAPÍTULO II

11 A nave espacial condenada

CAPÍTULO III

29 Pondo em cena

CAPÍTULO IV

35 Como é escrita uma aventura

CAPÍTULO V

39 Posição e Movimentos

CAPÍTULO VI

47 O computador entende português

CAPÍTULO VII

51 Plano, dicionário e objetos

CAPÍTULO VIII

59 Ação!

CAPÍTULO IX

67 Algumas ações

CAPÍTULO X

73 A bordo do "Neutronia"

CAPÍTULO XI

83 Últimos detalhes

CAPÍTULO XII

91 O intérprete

CAPÍTULO XIII

99 Variações sobre o mesmo tema

APÊNDICE A

105 Variáveis principais do intérprete

APÊNDICE B

107 Síntese de ação e prioridades de busca

PREFÁCIO

Neste livro pretendemos que todos acabem dominando a "arte" do desenho e programação de jogos em seu computador. Se não sabem o que é um jogo de aventuras poderão reparar essa grave lacuna intelectual de sua formação lendo simplesmente o capítulo 1 deste livro. Se já estão informados (ou são aventureiros experts) poderão saltar o capítulo ou então ler o mesmo para enriquecerem sua bagagem cultural com novas informações. De qualquer maneira, a escolha é sua: os leitores são vocês.

Também nos propomos, de início, insistir um pouco sobre o BASIC e o "estilo" de programação. "Ludendo docere" (ensinar jogando) era a expressão que poderia ter sido usada muitos séculos: isto é o que tentamos, dada nossa aversão para qualquer forma de ensinamento inutilmente enfadonha e "professoral", sem vontade de querer ofender (ao contrário, com todos os respeitos) aos verdadeiros professores, aqueles que sabem ativar o mecanismo mais importante e necessário para aprender: o interesse.

Que desfrutem!

CAPÍTULO I

A AVENTURA DOS JOGOS DE AVENTURAS

As origens

Em uma data que não conseguimos saber com exatidão, mas que de todas as formas é anterior à existência dos computadores pessoais (mais ou menos nos primeiros anos da década de setenta), Crowther e Woods inventaram uma nova modalidade de jogo, no qual o computador descreve um ambiente, ou um situação, com frases como esta (traduzimos livremente):

Estás em uma esplêndida habitação de uns dez metros de altura.
Os muros são rios congelados de cor alaranjada. Um caminho íngreme e uma passagem fácil saem dos lados leste e oeste da habitação.

Existe um alegre passarinho piando.

O jogador, que se identifica como o personagem que está investigando o misterioso lugar, pode comunicar-se com o computador por meio de frases simples escritas no teclado, por exemplo:

Captura o passarinho.

Obtendo como resposta:

O passarinho não tinha medo quando você entrou, mas ao

aproximar-se inquietou-se e não consegues capturá-lo.

Desta resposta, o jogador obtém algumas informações: por exemplo; que o passarinho tem medo dele, ou então, tem medo de algum objeto que ele possui. Efetivamente, também é possível pegar ou deixar diferentes objetos (a finalidade do jogo é encontrar todos os tesouros e sair intacto do lugar) e transferir-se de um lugar a outro.

O computador se comporta como se "entendesse" as frases do jogador, dando a impressão de que realmente "vive" as situações descritas.

Crowter e Woods chamaram a seu jogo simplesmente "Adventure", isto é, "Aventura". Um nome muito apropriado que desde então serve de referência a todos os jogos do mesmo tipo.

Desde os "MINIS" aos computadores pessoais

Ainda que o programa tenha sido escrito em um minicomputador, ou seja, uma máquina relativamente pequena para a época (concretamente um DEC PDP-10), o Adventure original certamente não estava destinado ao gandre público; naqueles tempos (que parecem tão distantes) os computadores eram máquinas caríssimas, somente acessíveis a poucos especialistas e patrimônio de grandes indústrias, universidades e centros de pesquisa.

Ainda assim, o jogo foi um grande êxito entre os poucos afortunados que puderam jogar com ele. Também foi causa de muitos problemas nos centros de trabalho, porque os diretores viam como os programadores se distraiam com problemas que tinham muito pouco que ver com seu trabalho. Segundo parece, todas as proibições foram inúteis: a única possibilidade era esperar que o jogo acabasse, então o trabalho podia ser relembrado.

Com a introdução dos computadores pessoais vários programadores se lançaram à tarefa de criar jogos de aventuras que funcionassem nas novas máquinas. O assunto não era nada fácil: basta pensar que o Adventure original (escrito em linguagem FORTRAN) ocupava quase 300 kbytes, enquanto que nos computadores pessoais haviam disponíveis somente 16 kbytes, mais ou menos.

Um dos precurssores dos jogos de aventuras nos computadores pessoais foi Scott Adams, que já em 1978 escreveu (em BASIC) um intérprete que lhe permitiu criar pequenas mas bem organizadas aventuras (explicaremos o que é um intérprete dentro de alguns capítulos).

Mais adiante, a evolução dos pequenos computadores permitiu escrever aventuras cada vez mais complexas, inclusive versões do Adventure original (para o Apple II, por exemplo, existem duas: Apple Adventure e Microsoft Adventure, praticamente idênticas).

Texto ou gráficos?

O diálogo escrito não é o único meio possível de comunicação entre jogador e computador: um ambiente ou uma situação também podem ser desenhados e não somente descritos. Portanto, existe também um grande número (e em constante aumento) de aventuras gráficas; existe uma espécie de disputa entre os partidários dos dois tipos de jogos. Pessoalmente, acreditamos que as duas formas têm suas coisas boas: se bem que uma imagem a cores é mais atraente que um texto escrito, também é verdade que o segundo estimula mais a fantasia; nenhum desenho nunca estará a altura de uma cena imaginada com base a uma descrição sugestiva. É a mesma diferença que existe entre um bom livro e um bom filme.

Fazemos notar que dissemos "um bom livro" e "um bom filme". Entendemos que ambos os tipos são válidos se estão bem feitos, mas existe uma verdadeira avalanche de aventuras decididamente péssimas (de ambos os gêneros). Retomaremos este tema mais adiante para tentar explicar os fatores que contribuem à qualidade de uma aventura.

Neste livro ensinaremos a construir aventuras de tipo texto, mas os mesmos princípios (como veremos) são aplicáveis perfeitamente à construção de aventuras gráficas.

CAPÍTULO II

A NAVE ESPACIAL CONDENADA

Introduzindo-nos no programa

Não é nossa intenção aborrecê-los com confusas propostas teóricas. Ao contrário, propomos em seguida uma aventura. Nos próximos capítulos mostraremos como nasceu este jogo, passo a passo. Aprenderão assim a desenhar "sob medida" suas próprias aventuras, aproveitando, se desejarem, uma parte (a mais complexa) do programa que propomos, e acrescentando os frutos de sua fantasia. Finalmente, poderão ver como funciona o programa e como modificá-lo segundo suas exigências particulares. Haverá também uma certa dose de teoria, porém diluída e (esperamos) digerível com facilidade. Este livro requer um conhecimento pelo menos superficial do BASIC. Se não o tem, ou se quiserem melhorará-lo, podem recorrer aos livros 5, 6, 7 e 9 de nossa coleção.

A listagem da aventura espacial aparece ao final deste capítulo.

A listagem que publicamos está em versão Apple II, mas para muitos outros computadores incluimos as variações necessárias. Uma única exceção importante: a versão para o Spectrum, que utiliza um BASIC não standard e bastante limitado.

Adaptação

- Apple II (todas as versões); nenhuma variação;

- Commodore 64: tirar o ponto de interrogação nos INPUT das linhas 800 e 1340 e substituir as linhas 1990-2090 pelas que aparecem na figura 1.
- MSX, IBM, Olivetti e Macintosh com microsoft BASIC: tirar o ponto de interrogação nos INPUT das linhas 800 e 1340, substituir a linha 1130 por:

```
1130 A$ = INPUT$ (1)
```

e substituir as linhas 1990-2090 pelas que aparecem na figura 2.

```
1990 REM 5:SAVE
2000 OPEN 1,1,1,F$
2010 FOR I=1 TO FO:PRINT#1,LO%(I):NEXT I
2020 PRINT#1,LU:PRINT#1,T1:PRINT#1,V1:PRINT#1,V2
2030 CLOSE 1:RETURN
2050 REM 6:LOAD
2060 OPEN 1,1,0,F$
2070 FOR I=1 TO FO:INPUT#1,LO%(I):NEXT I
2080 INPUT#1,LU,T1,V1,V2
1090 CLOSE 1:RETURN
```

Figura 1. — Subrotinas de "Save" e "Load" para o Commodore 64.

```
1990 REM 5:SAVE
2000 OPEN F$ FOR OUTPUT AS #1
2010 FOR I=1 TO FO:PRINT#1,LO%(I):NEXT I
2020 PRINT#1,LU:PRINT#1,T1:PRINT#1,V1:PRINT#1,V2
2030 CLOSE 1:RETURN
2050 REM 6:LOAD
2060 OPEN FOR INPUT AS #1
2070 FOR I=1 TO FO:INPUT#1,LO%(I):NEXT I
2080 INPUT#1,LU,T1,V1,V2
1090 CLOSE #1:RETURN
```

Figura 2. — Subrotinas de "Save" e "Load" para as últimas versões de BASIC da Microsoft.

Pode haver problema com as letras minúsculas. Se seu computador não as tem ou querem evitar qualquer problema, simplesmente escrevam tudo em maiúsculas. Se, ao contrário, têm o Commodore 64, se arriscam a ter um problema monstruoso que o Commodore arrasta há anos: trabalhando em modo "só maiúsculas" os códigos de caracteres são trocados na tela (e ficam inalterados no teclado). Se querem utilizar o C64 em modo "maiúsculas/minúsculas" devem:

- colocar o modo "maiúsculas/minúsculas" com SHIFT/C =
- escrever o programa em minúsculas,
- escrever o que está entre aspas tal e como aparece na listagem, com as seguintes exceções:

  — linhas 1350, 1360: "S" e "N" vão em minúscula,
  — linha 830: "R" vai em minúscula,
  — linhas 3990-4090: tudo em maiúsculas,
  — linhas 4090, 4230, 4360, 4510: "FD", "FP", "FA", "FO" em minúsculas. O mesmo nas linhas 1040, 1050, 1060, 1070.

Ao introduzir o programa tenham cuidado com uma série de coisas:

— não confundam a letra O (mais quadrada nas listagens) com a cifra zero,
— não omitam os espaços nas cadeias (os que existem entre aspas),
— não esqueçam os dois pontos, vírgulas ou pontos e vírgulas.

Se quiserem, podem economizar as linhas de comentário (as que começam com REM). Quando acabarem, comprovem bem tudo com a ajuda de outra pessoa.

A jogar!

Após introduzirem o programa na máquina (recordem-se de realizar uma cópia), podem dar o RUN e jogar. Não aconselhamos continuar com a leitura do livro antes de ter jogado, pois encontrariam as soluções da aventura, atrapalhando-lhes a diversão, e teriam mais dificuldades para compreender as contínuas referên-

cias à mesma contidas no programa.

Para os não experts em aventuras, eis aqui as instruções fundamentais para um jogo deste tipo: Vocês estão divididos em uma dupla personalidade, que poderíamos chamar o braço e a mente.

Devem dar ordens a "outro você mesmo" que vive dentro do computador e que tem que enfrentar o problema de uma nave espacial condenada a um final seguro. Podem fazê-lo com frases do tipo:

OLHA O CURINGA

(recordem que devem chamar-se sempre de tu) ou então indicando a direção na qual querem mover-se, por exemplo:

NORTE

ou, mais simplesmente:

N

As possíveis direções são NORTE, SUL, ESTE, OESTE, ACIMA e ABAIXO; abreviados ficam, respectivamente, como N, S, E, O, A, B. Provem as mesmas em todos os lugares e desenhem um bom plano.

Uma ação fundamental é OLHA, que permite ações que facilitam informação adicional. Outras ações essenciais são PEGUA e DEIXA, para manipular os objetos.

Para obter a lista dos objetos que se possui, escrevam COISA ou INVENTÁRIO.

Se querem gravar a situação em disco (ou cassete) para continuar mais adiante, escrevam:

SAVE

Esta ordem também é útil antes de executar uma opção perigosa, para não ter que iniciar desde o princípio. Se ocorrer algum acidente desagradável, basta escrever depois:

LOAD

Para retomar a situação exatamente como estava no momento do último SAVE.

Naturalmente, os vocabulários citados não esgotam todas as possibilidades. Se verdadeiramente não encontram a ação que desejam, podem dar uma olhada nas linhas 3990 a 4090 que contém as palavras admitidas.

A aventura é muito simples de compreender (restanto talvez a ação final que requer um pouco de racicínio), porém muito cuidada e "realista". Esperamos que se divirtam jogando com ela e que a resolvam sem muitos problemas. Em qualquer caso, não se desesperem: no próximos capítulos a descrevemos detalhadamente. Claro que seria melhor se a conseguissem resolver sozinhos...

Este é o programa em versão para o Apple II (para os outros computadores, veja o item anterior):

```
100  REM ######################
110  REM # AVENTURA : A NAVE #
115  REM #ESPACIAL CONDENADA #
120  REM ######################
130  REM
140  HOME : GOTO 710: REM   MAIN
150  REM
160  REM  PROCURA P$ NO DICIONARIO, C=CODIGO (0=AUSENTE)
170  REM
180 I = 1:F = FD
190 A =  INT ((I + F) / 2):A$ = DZ$(A): IF P$ = A$ THEN C = DZ(A): GOTO
    240
200  IF P$ > A$ THEN I = A + 1
210  IF P$ < A$ THEN F = A - 1
220  IF I <  = F THEN 190
230 C = 0
240  RETURN
250  REM
260  REM  EXTRAI P$ DE IN$(IN), ENCONTRA CODIGO C, SALTA ART.
270  REM
280 C = 0
290 C$ =  MID$ (IN$,IN,1): IF C$ = " " OR C$ = "'" THEN IN = IN + 1: GOTO
    290
300  IF IN > LI THEN P$ = "": GOTO 360
310 A = IN: REM  INICIO
320 C$ =  MID$ (IN$,IN,1)
```

```
330  IF C$ < > " " AND C$ < > "'" AND IN < = LI THEN IN = IN + 1: GOTO
     320
340 P$ =  MID$ (IN$,A,IN - A): GOSUB 180: REM  PROCURA
350  IF C = 7 THEN 280: REM    ARTIGO
360  RETURN
370  REM
380  REM  PROCURA ACAO A, A=ACAO (0 SE NAO ENCONTRADA)
390  REM
400 I = 1:F = FA:N = A
410 A =  INT ((I + F) / 2):M = CA(A): IF N = M THEN A = AZ%(A): GOTO 460

420  IF N > M THEN I = A + 1
430  IF N < M THEN F = A - 1
440  IF I < = F THEN 410
450 A = 0
460  RETURN
470  REM
480  REM  EXECUTA ACAO A, A=0 SE NAO ENCONTRADA
490  REM
500  GOSUB 400: IF A = 0 THEN 550
510  IF C2 = 0 OR A > 0 THEN 530: REM      SO VERBO OU NAO TESTA
520 A =  - A: IF OB = 0 THEN  PRINT "-AQUI NAO ESTA ": GOTO 540
530  GOSUB 1700: REM   EXECUTA
540 A = 1
550  RETURN
560  REM
570  REM   NUMERA OBJETOS NO LUGAR L, COM PREFIXO P$
580  REM
590  FOR I = 1 TO FO
600  IF  ABS (LO%(I)) = L THEN  PRINT P$;OB$(I);"."
610  NEXT I: RETURN
620  REM
630  REM  OB=INDICE OBJETO C2, 0 SE NAO PRES. 0 TRANSP.
640  REM
650 OB = 0: FOR I = 1 TO FO
660  IF CO%(I) = C2 THEN  IF  ABS (LO%(I)) = LU OR LO%(I) = 0 THEN OB =
     I:I = FO
670  NEXT I: RETURN
680  REM
```

```
690  REM  PRINCIPAL (PARSER)
700  REM
710  GOSUB 970: REM  INICIO
720  GOSUB 1540: REM  INTRODUCAO
730  GOSUB 1120: REM  <SP>
740  REM
750  REM  CICLO DE JOGO
760  PRINT : PRINT "Estas ";DE$(LU);".": REM   DESCRICAO
770 L = LU:P$ = "Vejo ": GOSUB 590: REM    OBJETOS
780  GOSUB 1400
790  PRINT : PRINT
800 IN$ = "": INPUT "O que tenho que fazer?";IN$: IF IN$ = "" THEN 800
810  PRINT :LI =  LEN (IN$):IN = 1: GOSUB 280:P1$ = P$:C1 = C
820  IF P1$ = "" THEN  PRINT "E entao?": GOTO 760
830  IF  RIGHT$ (P1$,1) = "R" THEN  PRINT "Me chame de voce, por favor":
     GOTO 760
840  IF C1 = 0 AND P1$ <  > "" THEN  PRINT "Nao conheco o verbo ";P1$: GOTO
    760
850  GOSUB 280:P2$ = P$:C2 = C
860  IF C2 = 0 AND P2$ <  > "" THEN  PRINT "Nao conheco a palavra ";P2$:
     GOTO 760
870  IF C2 <  > 0 THEN  GOSUB 650: REM  OB=INDICE
880 N1 = LU * 10000:N2 = C1 * 100
890 A = N1 + N2 + C2: GOSUB 500: IF A GOTO 760: REM         VERBO+NOME EM
     LU
900  IF C2 <  > 0 THEN A = N1 + N2 + 99: GOSUB 500: IF A GOTO 760: REM
      VERBO+X EM LU
910 A = N2 + C2: GOSUB 500: IF A GOTO 760: REM     VERBO+NOME GERAL
920  IF C2 <  > 0 THEN A = N2 + 99: GOSUB 500: IF A GOTO 760: REM     VE
    RBO+X GERAL
930  PRINT "Nao entendo": GOTO 760
940  REM
950  REM  INICIO
960  REM
970  DIM DZ$(100),DZ(100): REM  DICIONARIO
980  DIM DE$(30),DI$(30): REM       PLANO
990  DIM CA(150),AZ%(150): REM   ACOES
1000  DIM OB$(50),CO%(50),LO%(50): REM  OBJETOS
1010 F$ = "ASTRO": REM  ARQUIVO SITUACAO
```

```
1020  REM  FD,FP,FA,FO=0
1030  PRINT : PRINT "Um pouco de paciencia";
1040  READ A$: IF A$ <  > "FD" THEN FD = FD + 1:DZ$(FD) = A$: READ DZ(FD
    ): GOTO 1040
1050  READ A$: IF A$ <  > "FP" THEN FP = FP + 1:DE$(FP) = A$: READ DI$(F
    P): GOTO 1050
1060  READ A$: IF A$ <  > "FA" THEN FA = FA + 1:CA(FA) =  VAL (A$): READ
    AZ%(FA): GOTO 1060
1070  READ A$: IF A$ <  > "FO" THEN FO = FO + 1:OB$(FO) = A$: READ CO%(F
    O),LO%(FO): GOTO 1070
1080  PRINT : PRINT : RETURN
1090  REM
1100  REM  <SP> PARA CONTINUAR
1110  REM
1120  PRINT "Aperte espaco para continuar"
1130  GET A$: IF A$ <  > " " THEN 1130
1140  PRINT : PRINT : RETURN
1150  REM
1160  REM  ##FINAL INTERPRETE, PRINCIPIO AVENTURA##
1170  REM
1180  REM  MORTO
1190  REM
1200  GOSUB 1120: REM  <SP>
1210  PRINT "*** O BOLETIM DA GALAXIA ***"
1220  PRINT : PRINT "Tragedia ao redor de Vega": PRINT
1230  PRINT "A nave espacial Neutronia,em servico de"
1240  PRINT "passageiros, com 250 pessoas a bordo, "
1250  PRINT "foi destruida por uma violenta "
1260  PRINT "explosao, causada provavelmente"
1270  PRINT "pela impericia de seu comandante (um"
1280  PRINT "principiante, segundo rumores"
1290  PRINT "recolhidos pelo nosso correspondente)."
1300  PRINT "O responsavel do pior desastre"
1310  PRINT "da historia galatica vera sua memoria"
1320  PRINT "marcada pela tragedia": PRINT
1330  REM
1340  INPUT "Quer continuar jogando?";A$
1350 A$ =  LEFT$ (A$,1): IF A$ <  > "S" AND A$ <  > "N" THEN 1330
1360  IF A$ = "S" THEN  RUN
```

```
1370  PRINT : PRINT : PRINT "Adeus!!!": PRINT : END
1380  REM
1390  REM  TEMPO
1395  REM
1400 T1 = T1 - 1
1410  IF T1 <  = 10 THEN  PRINT "Ouco um sinal de alarme"
1420  IF T1 <  = 5 THEN  PRINT "A temperatura e insuportavel"
1430  IF T1 <  = 2 THEN  PRINT "A nave e sacudida por vibracoes"
1440  IF T1 > 0 THEN  RETURN
1450  PRINT : PRINT "*** Muito tarde !! ***": PRINT
1460  PRINT "O reator esta fora de controle"
1470  PRINT "A nave esta desintegrada"
1480  PRINT "em fragmentos minusculos."
1490  PRINT "E incrivel o silencio"
1500  PRINT "das explosoes no vacuo": GOTO 1200
1510  REM
1520  REM  INTRODUCAO
1530  REM
1540  PRINT : PRINT : PRINT
1550  PRINT "*******************************"
1560  PRINT "* A NAVE ESPACIAL CONDENADA *"
1570  PRINT "*******************************": PRINT
1580  PRINT "Sepultado na areia, desfruta do"
1590  PRINT "doce calor do sol tropical..."
1600  PRINT : PRINT "Agora o sol fica mais forte."
1610  PRINT "Estas em pleno deserto e nao tem"
1620  PRINT "nem sinal de oasis..."
1630  PRINT : PRINT "Te despertas sobressaltado em sua"
1640  PRINT "cabine de comandante do Neutronia"
1650  PRINT "Faz calor. Muito calor. Tem"
1660  PRINT "que haver algo que nao funciona."
1670 T1 = 100:LU = 6:V1 = 0:V2 = 0: PRINT : RETURN
1680  REM
1690  REM  EXECUTA A ACAO A
1700  ON A GOTO 1780,1830,1920,1970,2000,2060,2120,2150,2180,2210
1710  ON (A - 10) GOTO 2260,2300,2380,2410,2460,2580,2640,2700,2740,2800
1730  ON A - 20 GOTO 2860,2910,2960,3010,3080,3190,3320,3370,3400
1740  ON A - 30 GOTO 3440,3520,3570,3640,3800,3860,3900,3940
1750  PRINT "ACAO";A: RETURN : REM  TESTE
```

```
1760  REM
1770  REM  REM 1 DIRECOES
1780 A =  VAL ( MID$ (DI$(LU),2 * C1 - 1,2))
1790  IF A = 0 THEN  PRINT "Por ali nao pode ir": RETURN
1800 LU = A: RETURN
1810  REM
1820  REM   2  PEGAR
1830  IF LO%(OB) = 0 THEN  PRINT "JA FEITO": RETURN
1840  IF LO%(OB) < 0 THEN  PRINT "Nao e possivel": RETURN
1850  IF OB = 4 AND LO%(22) = 0 THEN  PRINT "Tire antes o traje": RETURN

1860  IF O = 22 AND LO%(4) = 0 THEN  PRINT "Tire antes o macacao": RETURN

1870 LO%(OB) = 0
1880  IF OB = 4 OR OB = 9 OR OB = 11 THEN  PRINT "Agora o tens posto": RETURN

1890  PRINT "Feito": RETURN
1900  RETURN
1910  REM  3 DEIXA
1920  IF OB = 0 OR LO%(OB) <  > 0 THEN  PRINT "Nao o tens": RETURN
1930  IF LU < 9 THEN LO%(OB) = LU: PRINT "Feito": RETURN
1940 LO%(OB) =  - 99: PRINT "Se perdeu no espaco": RETURN
1950  REM
1960  REM  4 OLHA
1970  PRINT "Nao noto nada de particular": RETURN
1980  REM
1990  REM  5 SAVE
2000 D$ =  CHR$ (4): PRINT D$;"OPEN";F$: PRINT D$;"WRITE";F$
2010  FOR I = 1 TO FO: PRINT LO%(I): NEXT I
2020  PRINT LU: PRINT T1: PRINT V1: PRINT V2
2030  PRINT D$;"CLOSE": RETURN
2040  REM
2050  REM  6 LOAD
2060 D$ =  CHR$ (4): PRINT D$;"OPEN";F$: PRINT D$;"READ";F$
2070  FOR I = 1 TO FO: INPUT LO%(I): NEXT I
2080  INPUT LU,T1,V1,V2
2090  PRINT D$;"CLOSE": RETURN
2100  REM
2110  REM  7 COISA
```

```
2120  PRINT "Possues:":L = 0:P$ = "- ": GOTO 590
2130  REM
2140  REM  8
2150  RETURN
2160  REM
2170  REM  9
2180  RETURN
2190  REM
2200  REM  10
2210  IF LO%(OB) <  > 0 THEN  PRINT "Nao o tens": RETURN
2220  IF  NOT (LU >  = 9 OR (LU = 7 AND V2 = 1)) THEN 1920: REM  DEIXA
2230  PRINT "O ar, o ar!!  Aaaagh!!!": GOTO 1200
2240  REM
2250  REM  11
2260  PRINT "E seu macacao para atividades"
2270  PRINT "extraveiculares": RETURN
2280  REM
2290  REM  12
2300  PRINT "Esta sem sentido e tem"
2310  PRINT "um amolgamento no capacete."
2320  PRINT "Provavelmente foi atingido"
2330  PRINT "por um meteorito enquanto"
2340  PRINT "reparava a antena. Por"
2350  PRINT "sorte ainda esta vivo": RETURN
2360  REM
2370  REM  13
2380  PRINT "Nao e melhor o ler?": RETURN
2390  REM
2400  REM  14
2410  PRINT "E bastante pesado"
2420  PRINT "Provavelmente esta"
2430  PRINT "tratado com chumbo": RETURN
2440  REM
2450  REM  15
2460  IF LO%(OB) = LU THEN  PRINT "Pegue-o antes": RETURN
2470  PRINT "-MANUAL DE INSTRUCOES DO-"
2480  PRINT "-   REATOR POSITRONICO  -"
2490  PRINT "-     MOD.YTREWQ 8421   -": PRINT
2500  PRINT "Para ativar o reator"
```

```
2510  PRINT "puxar a alavanca e depois apertar"
2520  PRINT "consecutivamente os pulsadores verde,"
2530  PRINT "amarelo e vermelho."
2540  PRINT "Para desativar o reator..."
2550  PRINT : PRINT "Maldicao!! A pagina esta arrancada": RETURN
2560  REM
2570  REM  16
2580  PRINT "-TEMPERATURA REATOR-": PRINT
2590  PRINT "Marca ";840 - T1 * 4;" graus e esta"
2600  PRINT "subindo rapidamente. Tem"
2610  PRINT "uma marca vermelha em 800 graus": RETURN
2620  REM
2630  REM  17
2640  PRINT "S.O.S. GALATICO": PRINT
2650  PRINT "Apertar o pulsador somente"
2660  PRINT "em caso de emergencia"
2670  PRINT "Todo abuso sera castigado": RETURN
2680  REM
2690  REM  18
2700  PRINT "Acende brevemente um aviso": PRINT
2710  PRINT "ANTENA EXTERIOR DEFEITUOSA": RETURN
2720  REM
2730  REM  19
2740  PRINT "Esta na base de uma escadinha"
2750  PRINT "e diz:": PRINT
2760  PRINT "ENTRADA RESERVADA AO"
2770  PRINT "PESSOAL DE BORDO": RETURN
2780  REM
2790  REM  20
2800 LU = 1: IF LO%(20) <  > 9 THEN  RETURN
2810  PRINT "Oxala astivesse aqui o segundo"
2820  PRINT "piloto, e o unico que entende"
2830  PRINT "os problemas tecnicos": RETURN
2840  REM
2850  REM  21
2860  PRINT "Melhor nao despertar os"
2870  PRINT "passageiros, poderiam"
2880  PRINT "deixar-se levar pelo panico": RETURN
2890  REM
```

```
2900  REM  22
2910  PRINT "Esta colocado a oeste e diz:": PRINT
2920  PRINT "ATENCAO:": PRINT
2930  PRINT "Habitacao despressurizada": RETURN
2940  REM
2950  REM  23
2960  IF LO%(24) <  > - 99 THEN  PRINT "Ja feito": RETURN
2970  IF LO%(23) <  > 0 THEN  PRINT "Esta fechado com chave": RETURN
2980  PRINT "Feito":LO%(24) = LU:LO%(22) = LU: RETURN
2990  REM
3000  REM  24
3010  PRINT "Esta vazio": RETURN
3020  REM
3030  REM  25
3040  PRINT "Aperta o vermelho"
3050  PRINT "ou aperta o verde": RETURN
3060  REM
3070  REM  26
3080  IF V2 = 1 THEN  PRINT "CLICK": RETURN
3090  PRINT "A parede a ESTE se fecha."
3100  PRINT "A parede a OESTE se abre"
3110  PRINT "ate o espaco exterior."
3120  PRINT "O ar sai assobiando."
3130  IF LO%(4) <  > 0 OR LO%(11) <  > 0 THEN  PRINT : PRINT "Aaagh!!": GOTO
    1200
3140  FOR I = 1 TO FO
3150  IF LO%(I) = LU THEN  PRINT OB$(I);" Se perde no espaco":LO%(I) =  -
    99
3160  NEXT I:V2 = 1: RETURN
3170  REM
3180  REM  27
3190  IF V2 = 0 THEN  PRINT "CLICK": RETURN
3200  PRINT "A parede a OESTE se fecha."
3210  PRINT "A parede a ESTE se abre"
3220  PRINT "ate o corredor."
3230  PRINT "O ar volta a entrar assobiando.":V2 = 0
3240  IF LO%(20) <  > 0 OR LO%(23) <  > - 99 THEN  RETURN
3250  PRINT : PRINT "O segundo piloto revive, se da"
3260  PRINT "conta em seguida da situacao e diz:"
```

```
3270  PRINT : PRINT "Pronto, para o reator!"
3280  PRINT "Eis aqui a chave de me..."
3290  PRINT "Depois perde de novo os sentidos":LO%(23) = LU: RETURN
3300  REM
3310  REM  28
3320  PRINT "A emergencia acesa!"
3330  PRINT "Leva sempre contigo"
3340  PRINT "o manual do reator": RETURN
3350  REM
3360  REM  29
3370  PRINT " O tecnico e o segundo piloto": RETURN
3380  REM
3390  REM  30
3400  PRINT "Aperta o vermelho, aperta"
3410  PRINT "o verde ou aperta o amarelo": RETURN
3420  REM
3430  REM  31
3440  PRINT "CLICK": IF V1 <  > 0 THEN 3830
3450  PRINT "Uma tubulacao vaza ligeiramente"
3460  PRINT "por uma juncao (provavelmente por"
3470  PRINT "excesso de pressao).As gotas caem sobre"
3480  PRINT "o quadro de controle, perto de voce."
3490 V1 = 1: RETURN
3500  REM
3510  REM  32
3520  PRINT "CLICK": IF V1 <  > 1 THEN 3830
3530 V1 = 2: IF LO%(22) <  > 0 THEN  PRINT "Nao  te encontras muito bem"
3540  RETURN
3550  REM
3560  REM  33
3570  PRINT "CLICK": IF V1 <  > 2 THEN 3830
3580 V1 = 3: IF LO%(22) = 0 THEN  RETURN
3590  PRINT "Temo que tenhas absorvido muita"
3600  PRINT "radiacao. Agora estas realmente"
3610  PRINT "mal. Perde o sentido...": GOTO 1200
3620  REM
3630  REM  34
3640  PRINT "CLANK": IF V1 <  > 3 THEN 3830
3650  GOSUB 1120: REM  <SP>
```

```
3660  PRINT "*** O BOLETIM DA GALAXIA ***"
3670  PRINT : PRINT "COMANDANTE SALVA NAVE ESPACIAL"
3680  PRINT "A nave espacial Neutronia, em servico"
3690  PRINT "de passageiros,com 250 pessoas a bordo,"
3700  PRINT "foi salva da certa destruicao"
3710  PRINT "gracas ao valor e ao sangue frio"
3720  PRINT "do comandante que conseguiu"
3730  PRINT "desativar o reator enlouquecido."
3740  PRINT "Seu nome sera recordado"
3750  PRINT "sempre entre os herois de nossa"
3760  PRINT "frota galatica": PRINT
3770  PRINT : PRINT "FELICIDADES!!": PRINT : END
3780  REM
3790  REM  35
3800  PRINT "CLUNK": GOTO 3830
3810  REM
3820  REM   COMUM
3830 V1 = 0:T1 =  INT (T1 / 2): RETURN
3840  REM
3850  REM  36
3860  PRINT "Parece avariada"
3870  PRINT "por um grande meteorito": RETURN
3880  REM
3890  REM  37
3900  IF V2 = 0 THEN LU = 4: RETURN
3910  GOTO 1780: REM  DIRECAO
3920  REM
3930  REM  38
3940  IF V2 = 1 THEN LU = 11: RETURN
3950  GOTO 1780: REM  DIRECAO
3960  REM
3970  REM  DICIONARIO
3980  REM
3990  DATA "A",5,"A/",7,"ABAIXO",6,"ABRE",22,"ACIMA",5
4000  DATA "AJUSTA",27,"ALAVANCA",65,"AMARELO",63,"ANTENA",69,"APERTA",2
    6
4010  DATA "ARMARIO",67,"B",6,"BAIXA",6,"CAMA",68,"CAPACETE",50
4020  DATA "CARTAZ",60,"CHAVE",54,"COISA",13,"DEIXA",9
4030  DATA "E",3,"EMPURRA",24,"ESTE",3,"ETIQUETA",70
```

```
4040  DATA "INDICADOR",66,"INVENTARIO",13,"LE",25,"LOAD",12
4050  DATA "MACACAO",51,"MANUAL",55,"N",1,"NORTE",1
4060  DATA "O",4,"O/",7,"OESTE",4,"OLHA",10
4070  DATA "PEGA",8,"PONHA",20,"PULSADOR",61,"PUXA",23,"REPARA",27
4080  DATA "S",2,"SAVE",11,"SEGUNDO",53,"SOBE",5,"SUL",2,"TECLA",61
4090  DATA "TIRA",21,"TRAJE",52,"VERDE",62,"VERMELHO",64,"VESTE",20,"W",
   40,"FD"
4100  REM
4110  REM  PLANO
4120  REM
4130  DATA "na cabine de comando","000000000002"
4140  DATA "em um extremo do corredor","000300050100"
4150  DATA "no corredor","020400060000"
4160  DATA "em um extremo do corredor","030000070008"
4170  DATA "na cabine do segundo piloto","000002000000"
4180  DATA "em sua cabine","000003000000"
4190  DATA "no compartimento estanque","000000000000"
4200  DATA "na sala do reator","000000000400"
4210  DATA "no exterior, a proa da nave","001000000000"
4220  DATA "no exterior da nave","091100000000"
4230  DATA "no exterior a popa da nave","100007000000","FP"
4240  REM
4250  REM   ACOES
4260  REM
4270  DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2
4280  DATA 950,-10,951,-10,999,3,1051,-11,1052,-14,1053,-12
4290  DATA 1055,-13,1060,-13,1099,-4,1100,5,1200,6,1300,7
4300  DATA 2050,-2,2051,-2,2052,-2,2150,-10,2151,-10,2152,-3,2255,-15
4310  DATA 2555,-15,2769,-29,6550,-10,6551,-10,6552,-3
4320  DATA 11066,16,12570,17,12661,18,20300,21,20500,20,22560,19
4330  DATA 30300,21,40300,21,42560,22,52267,23,62267,24
4340  DATA 70300,37,70400,38
4350  DATA 72661,25,72662,27,72664,26,81066,16,82365,35,82465,34
4360  DATA 82560,28,82661,30,82662,33,82663,32,82664,31,91069,36,"FA"
4370  REM
4380  REM  OBJETOS
4390  REM
4400  DATA "um indicador",66,-1,"um pulsador",61,-1
4410  DATA "uma etiqueta",70,-1,"um macacao",51,1
```

```
4420  DATA "um cartaz branco",60,-2,"um cartaz amarelo",60,-4
4430  DATA "uma cama",68,-5,"um armario",67,-5
4440  DATA "uma cama",68,-6,"um armario",67,-6
4450  DATA "um capacete",50,6,"um pulsador vermelho",61,-7
4460  DATA "um pulsador verde",61,-7,"um indicador",66,-8
4470  DATA "uma alavanca",65,-8,"um pulsador vermelho",61,-8
4480  DATA "um pulsador verde",61,-8,"um cartaz vermelho",60,-8
4490  DATA "um pulsador amarelo",61,-8,"o segundo piloto",53,9
4500  DATA "uma antena parabolica",69,-9,"um traje",52,-99
4510  DATA "uma chave",54,-99,"um manual",55,-99,"FO"
```

CAPÍTULO III

PONDO EM CENA

Atenção: este capítulo contém a solução da aventura. Se lerem antes de jogar perderão toda a diversão.

A idéia

Realizar uma aventura requer fantasia e técnica; igual a escrever uma novela, pintar um quadro ou qualquer outra forma de expressão, ambos os componentes são fundamentais para realizar um bom trabalho.

Está claro que **nós podemos somente proporcionar-lhes os instrumentos técnicos, mas não a fantasia e a criatividade, ainda que possamos ser-lhes úteis na organização das idéias e na fantasia inicial de criação do jogo**; ensinando-lhes como foi gestionado e como nasceu "A nave espacial condenada".

A primeira necessidade é uma idéia, uma ambientação: onde transcorre a aventura? Qual é o argumento da história?

Era um sábado pela manhã de um cruel inverno, com 85 centímetros de neve rodeando a casa...; mas não divaguemos, esta é outra aventura. Era sábado pela manhã, dizíamos, e estávamos discutindo sobre qual poderia ser o tema de uma pequena aventura para colocá-lo de exemplo neste livro. A primeira idéia foi a procura de um tesouro em uma ilha do Caribe, nas antigas ruínas dos piratas. Escrevemos algumas notas; as idéias não nos faltavam, inclusive algumas eram tão boas que decidimos guardá-las para ou-

tra futura aventura (isto acontece às vezes). Mas logo pensamos que a história da ilha não nos convencia; então que ambiente utilizar?

Não queríamos uma ambientação "gótica" (tipo "Aventura no castelo") porque não servia para aventuras de pequeno tamanho, onde a ambientação não pode ser desenvolvida mais além de um determinado ponto. Também era necessário motivar ao aventureiro: não existe nada pior que uma aventura onde o jogador se sente "fora" do jogo ao invés de "vivê-lo" realmente em primeira pessoa. Um reator nuclear enlouquecido? Muito simples. Um computador malicioso a bordo de uma nave espacial (como o HAL 9000 na célebre "2001, Uma odisséia no espaço")? Mais interessante, porém difícil de desenvolver em pouco espaço. Tiremos o computador e colocamos o reator enlouquecido a bordo da nave espacial. Quase chegamos. Acrescentamos um tempo limite e 250 passageiros, ignorantes do que acontece, que temos que salvar (além de nós mesmos). Sim, assim está melhor

O guia

Agora que já temos a idéia de partida, tentamos desenvolver um guia coerente. O protagonista é o comandante de uma nave espacial que terá que conseguir deter a tempo o reator. Naturalmente, terá que fazer tudo sozinho: não pode descarregar em ninguém nenhuma parte de sua própria responsabilidade pois, senão, o jogo perde interesse e é convertido em um "passar a bola" de uns a outros. Então, o comandante está só enquanto que os passageiros estão, colocamos assim, dormindo tranqüilamente.

Se bem que pode ser certo que as modernas naves espaciais requerem pouco pessoal, é pouco plausível que seja suficiente uma só pessoa. Se no jogo é comprovado que o comandante não é capaz de parar o reator, é necessário que tenha outra pessoa capaz de fazê-lo. Por outro lado, essa outra pessoa não deve estar muito a mão, porque senão o problema seria resolvido facilmente. Introduzimos então a figura de um segundo piloto, com funções técnicas, mas fazendo com que não esteja facilmente disponível. Em princípio não se sabe o que acontece com ele: poderia ter se suicidado (típico sintoma de depressão espacial), ter sido vítima de acidente (talvez por causa do reator) ou, simplesmente, andar por aí despistado.

Em toda aventura espacial que se preze não pode faltar uma saída ao espaço exterior. Portanto deveremos admití-la de algu-

ma maneira. Eis aqui como: o segundo piloto se encontra no exterior da nave espacial e somente resgatando-o poderá ser obtida uma informação assencial. Nesse momento ainda pode estar vivo: assim seu resgate fará parte da missão. Uma vez resgastado, recobra o sentido (mas somente por um momento) e nos entrega algo fundamental. O que? Vejamos... O livro de instruções que serve para desativar o reator. Não, alonguemos um pouco mais o assunto: a chave de seu armário, que contém o manual de instruções e um traje anti-radiações necessários para trabalhar no reator. Além disso, para dificultar a solução, tampouco estará muito claro o que é que nos dá.

A propósito, por que o segundo piloto estava no exterior? Utilizaremos uma idéia de "2001": estava ajustando a antena de comunicações (a nave espacial também estará, portanto, isolada do resto do universo).

Com isto o guia está completo em suas linhas gerais: é necessário equipar-se para sair ao vácuo (macacão e capacete), sair ao

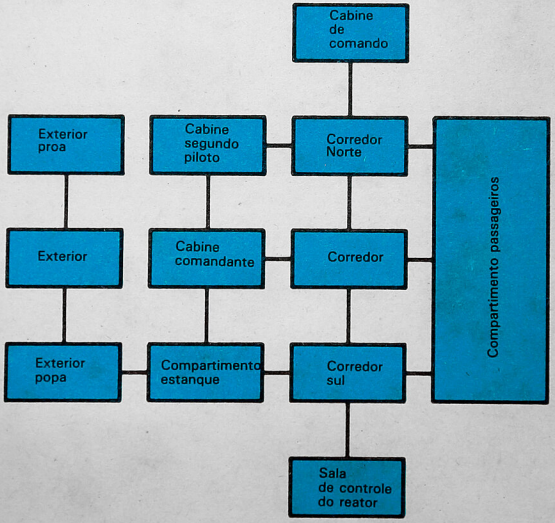

Figura 1. — Disposição de todos os lugares possíveis em "A nave espacial condenada".

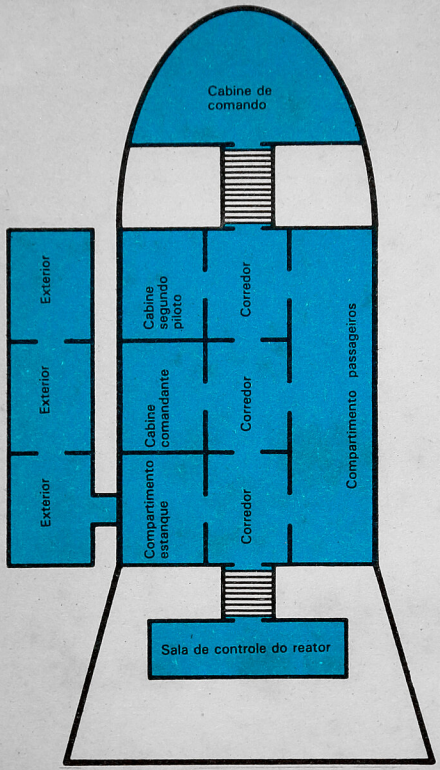

Figura 2. — Planta da nave espacial de passageiros "Neutronia".

exterior, resgatar ao segundo piloto, conseguir a chave, abrir seu armário, pegar o manual e o traje, ler o manual e desativar o reator.

Parece executável. Os detalhes serão acrescentados pouco a pouco.

O plano da nave

Nos falta estabelecer a disposição das diversas estâncias que configuram a nave espacial. Desenhamos um plano, que nos sirva de guia, com o mesmo sistema já usado em outras ocasiões: quadrados ligados com linhas que indicam os caminhos e direções possíveis. Adotando um critério convencional o Norte está acima, o Sul abaixo, o Este à direita e o Oeste à esquerda (como mapas geográficos). Para indicar um movimento para cima, fazemos partir a linha desde o ângulo superior direito e se é para baixo, desde o ângulo inferior esquerdo.

A Figura 1 mostra a disposição dos locais, desenhada com este sistema. Note o compartimento estanque, necessário para sair ao exterior sem que escape o ar da nave espacial.

Parece que está bem projetado, podemos passá-lo a limpo. A Figura 2 mostra a versão definitiva do mesmo plano. As proporções reais não foram respeitadas, e omitimos alguns locais de menor interesse, mas as severas Normas de Segurança Galática do 2312 nos proíbem proporcionar mais informação, ainda que se trate de uma nave espacial civil.

À parte dos detalhes, o desenho do projeto da aventura terminou: não resta mais que colocar mãos à obra. Como informação adicional que, à parte do tempo para passar a limpo, empregamos pouco mais de um par de horas.

CAPÍTULO IV

COMO SE ESCREVE UMA AVENTURA

Da idéia ao programa

Agora que já dispomos de um guia aceitável, podemos começar a escrever o programa. Como é feito um programa deste tipo? É o mesmo problema com o qual nos encontramos em 1981 quando, depois de ter jogado o "Apple Adventure", decidimos escrever uma aventura por nossa conta. Naquele tempo estávamos totalmente desinformados e não líamos sequer as principais revistas do setor, especialmente as americanas.

Uma pena, porque poderíamos, pelo menos, conhecer o que outros haviam feito ou estavam fazendo. Isto se explica porque naqueles tempos éramos simplesmente afeiçoados neste tema e essas publicações não circulavam nem sequer nos ambientes supostamente mais qualificados (por exemplo, a Universidade), onde contudo estava vigente o culto ao computador-dinossauro (à base de cartões perfurados e com longas caudas para poder usá-lo).

Não tivemos mais remédio que ir aprendendo pouco a pouco e ajustar-nos por nós mesmos. Por sorte tínhamos bastante prática em programação, de forma que ao final acabamos sabendo um pouco. Depois de um ano de trabalho nos momentos livres, conseguimos escrever um programa ao qual inclusive hoje, como profissionais, podemos olhar sem sentirmos vergonha. Dadas as circunstâncias, é uma grande satisfação. "A nave espacial condenada" deriva diretamente de "Aventura no castelo", que é o programa que acabamos de mencionar, com muitas ampliações e me-

lhoras derivadas de um estudo do qual falaremos posteriormente.

A melhora fundamental, pelo menos do nosso ponto de vista, consiste na linguagem utilizada: o programa "Aventura no castelo" (em sua primeira versão) estava escrito metade em BASIC, metade em Anssembler do 6502 (a linguagem máquina do Aplle), com todos os truques possíveis para economizar memória e aumentar a velocidade de execução. Isto significa que estava ligado indissoluvelmente ao modelo específico de computador no qual havia sido projetado. O programa da "A nave espacial condenada" está escrito todo em BASIC, e em um BASIC "limpo". Portanto é facilmente transferível, isto é, pode funcionar com mínimas modificações em computadores tão distintos entre si como um IBM PC e um Commodore 64. Ao contrário, não se presta muito à realização de aventuras de grandes dimensões: não se pode ter tudo. As simplificações que introduzimos (e que diminuem um pouco a "inteligência" do programa) foram feitas em grande parte para concentrar as explicações nos aspectos essenciais e não distrair com os floreados do programa.

Delineamentos equivocados

Dizíamos que ainda estávamos satisfeitos com o delineamento do nosso primeiro programa de aventuras. Estamos porque, olhando ao nosso redor, vemos que a grande maioria das aventuras estão construídas com técnicas muito mais primitivas e trabalhosas.

Também em alguns livros que ensinam como escrever uma aventura, existem delineamentos que obrigam ao programador a um trabalho excessivamente longo, grande parte do qual poderia ter sido tranqüilamente reduzido por um programa melhor pensado e mais flexível.

Esta é a mesma essência da programação dos computadores: escrever um programa que resolva não somente um problema, mas um conjunto de problemas parecidos modeláveis conforme variam os dados introduzidos. Não teria sentido escrever, por exemplo, um programa para somar 1 + 1, outro para somar 1 + 2, outro distinto para somar 2 + 2, e assim sucessivamente: um só programa pode resolver todos estes problemas. Parece óbvio neste caso, mas ao aumentar a complexidade e variabilidade do problema, o assunto é cada vez menos fácil e requer um estudo mais detalhado. A tentação de escrever um programa que resol-

va somente o problema atual é cada vez maior. Por exemplo, vimos um jogo de aventuras escrito mais ou menos deste modo:

```
500 REM MORADIA BRANCA
510 PRINT "ESTA NA MORADIA BRANCA. EXISTE UM ARMARIO"
520 INPUT A$
530 IF A$="N" OR A$="NORTE" THEN GOTO 610:REM VERMELHA
540 IF A$="E" OR A$="ESTE" THEN GOTO 710:REM VERDE
550 PRINT "NAO ENTENDO":GOTO 510
...
600 REM MORADIA VERMELHA
610 PRINT "ESTA NA MORADIA VERMELHA. EXISTE UM ARMARIO"
620 INPUT A$
630 IF A$="S" OR A$="SUL" THEN 510:REM BRANCA
640 IF A$="ABRE O ARMARIO" THEN PRINT "ESTA VAZIO":GOTO
610
650 PRINT "NAO ENTENDO":GOTO 610
```

Entendeu como funciona? Para cada lugar existe o correspondente grupo de instruções dos quais o programa continua rodando até que o aventureiro não decida mudar de lugar. Por exemplo, desde as linhas 500 a 550 é feita referência a um lugar chamado "Moradia branca". A linha 510 descreve o lugar e a 520 espera as ordens do jogador (tipicamente figurará uma subrotina em lugar de uma instrução INPUT, mas o conceito é o mesmo). Se o jogador escreveu "N" ou "NORTE", ambas as palavras são válidas para indicar um deslocamento para o norte, o programa salta ao grupo de instruções ao lugar que se encontra ao norte da moradia branca, no exemplo, uma hipotética moradia vermelha (linha 610). Se foi escrito "E" ou "ESTE", salta às instruções referentes à "Moradia verde", e assim sucessivamente. Se a ordem não está entre as previstas naquela moradia, a linha 550 volta ao princípio do loop (510). Cada moradia (lugar) tem seu própio ciclo de intruções, dentro do qual são admitidas somente determinadas palavras ou frases. Por exemplo, no ciclo da moradia vermelha (linhas 610-650) são admitidos "N" (ou "NORTE"), que volta ao ciclo da moradia branca e, "ABRE O ARMARIO", que imprime "ESTA VAZIO" e volta ao princípio do ciclo. A mesma explicação vale para todos os lugares nos quais o protagonista pode se mover. É uma técnica simples e compreensível.

Um delineamento deste tipo tem que ser evitado. Por que? Vamos ver uma lista de bons motivos para fazê-lo:

- o controle das direções possíveis (IF A$ = "N" ou A$ = NORTE"...) é repetido em cada loop de lugar;
- as frases válidas devem ser escritas exatamente como estão previstas no programa. Se a frase "ABRE O ARMARIO" é escrita com um espaço ao princípio ou ao final ou entre "ABRE" e "ARMÁRIO", é obtido um "NÃO ENTENDO". O mesmo acontece se o artigo não está bem colocado;
- existem notáveis problemas com as frases válidas para todos os lugares, por exemplo, as ações sobre um objeto transportável;
- se desejamos escrever outra aventura deveremos voltar a escrever o programa desde o princípio até o final.

Poderíamos continuar, mas nos parece suficiente. Em essência, é necessário um grande trabalho para obter um programa pouco satisfatório e que ocupa inutilmente muita memória. Pior ainda, o jogo se apresenta com um comportamento decididamente "estúpido", no sentido de que aceita exclusivamente as frases previstas no lugar no qual se encontra o aventureiro (que corresponde a um "lugar" físico do programa, entendido como um grupo de instruções).

O que lhes apresentamos é um exemplo limite (desgraçadamente real) porém grande parte das aventuras atuais apresentam problemas devidos a um errôneo delineamento do programa, que é traduzido posteriormente em uma perda de realismo e, portanto, de interesse do jogador. Comentário típico: - Mas, esta máquina estúpida não entende nada? - Já a ilusão de viver uma aventura fantástica foi perdida, junto com os esforços do autor.

Está claro que a qualidade de uma aventura depende de quem a tenha escrito, porém também está claro que a fantasia não é suficiente: sem as técnicas apropriadas é difícil realizar um jogo que seja capaz de "capturar" os jogadores proporcionando um nível aceitável de simulação. Nos próximos capítulos demonstraremos como é possível obter isto com uma estrutura compacta, fácil de usar e reutilizável para outras aventuras.

CAPÍTULO V

POSIÇÕES E MOVIMENTOS

A melhor solução não é um programa que admite as distintas instruções conforme o lugar, como mostramos (colocando-a como exemplo negativo) no capítulo anterior, mas a consistente em um programa que execute sempre o mesmo ciclo fundamental, mudando somente o valor de algumas variáveis e, conseqüentemente, as mensagens impressas na tela. Um exemplo deste delineamento foi apresentado pelo volume número 9 ("Programando como se deve") desta mesma coleção.

Uma subrotina universal

Para aqueles que não o leram e também, por que não?, para os demais, propomos um exemplo parecido: um mini-programa ou, melhor dito, uma subrotina (isto é, uma parte de programa) que imprime a descrição da moradia na qual se encontra o aventureiro e que lhe permite mover-se com o sistema habitual, ou seja, escrevendo a direção na qual quer mover-se. Eis aqui a subrotina:

```
100 REM INICIA VARIAVEIS
...
200 REM CICLO
210 PRINT DE$(LU):REM DESCRICAO
220 INPUT A$:D=0
```

```
230 IF A$="N" OR A$="NORTE" THEN D=1
240 IF A$="S" OR A$="SUL" THEN D=2
250 IF A$="E" OR A$="ESTE" THEN D=3
260 IF A$="O" OR A$="OESTE" THEN D=4
270 IF A$="A" OR A$="ACIMA" THEN D=5
280 IF A$="B" OR A$="ABAIXO" THEN D=6
290 IF D=0 THEN PRINT "NAO ENCONTRADO":GOTO 210
300 NL=PL (LU,D):REM NOVO LUGAR
310 IF NL=0 THEN PRINT "NAO PODE":GOTO 210
320 LU=NL:GOTO 210:REM MOVE
```

Isto é tudo. Se ao princípio (entre as linhas 100 e 200) introduzimos os dados apropriados nos elementos das matrizes DE$ e PL, esta subrotina, de 13 linhas, fará todo o trabalho necessário para controlar os movimentos do aventureiro sobre um terreno de qualquer complexidade ou dimensão. Não acreditam? Passemos em seguida a uma demonstração prática. Vamos supor que nosso mapa esteja composto por três lugares, chamados "Moradia vermelha", "Moradia branca" e "Moradia verde", dispostos como na Figura 1. A Figura 2 mostra como deve ser o conteúdo das matrizes DE$ e PL. A primeira, de tipo alfanumérico (cadeia), contém

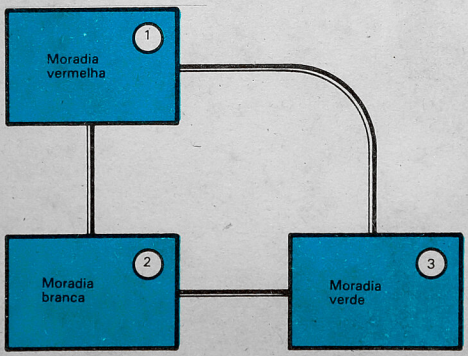

Figura 1. — Exemplo mínimo de um plano para desenhar aventuras.

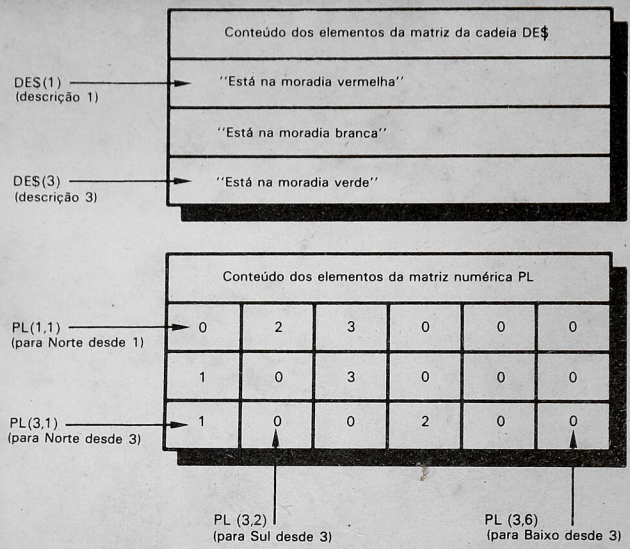

Figura 2. — Matrizes essenciais do programa e seu conteúdo.

simplesmente as descrições dos lugares em ordem numérica. Portanto DE$ (1) é a descrição do lugar 1, DE$(2) é a descrição do lugar 2 e DE$(3) a descrição do lugar 3. Como puderam deduzir, os lugares vão numerados a partir de 1. Voltamos por um momento ao programa e precisamente à linha 210. Esta imprime o conteúdo de DE$ (LU), isto é, a descrição do lugar número LU. LU contém, portanto, o número do lugar atual (ou seja, aquele no qual se encontra o aventureiro). Vamos supor que colocamos LU = 1 ao princípio do programa: isto significa que o personagem se encontra no lugar 1; assim a linha 210 imprime DE$ (1), isto é, "Está na habitação vermelha" que, efetivamente, é a descrição do lugar 1. Chamamos DE a matriz por "Descrição"; sempre é melhor utilizar para as variáveis nomes que ajudem a recordar seu significado.

Numeramos os lugares:

- 1 Moradia vermelha

- 2 Moradia branca
- 3 Moradia verde

Chegados aqui, terão compreendido que para deslocar o aventureiro, ou seja, para mover-se de lugar é suficiente mudar o valor de LU. Se LU vale 2, a linha 210 imprime "Está na moradia branca" e se vale 3 imprime "Está na moradia verde". O jogador tem a sensação de que muda de lugar, mas para o programa muda simplesmente o valor de uma variável, precisamente LU. As instruções executadas permanecem invariáveis.

De acordo, mas como fazemos para mudar LU segundo a palavra escrita pelo jogador, de forma que os deslocamentos sejam coerentes com nosso plano? É muito simples: a cada lugar foram atribuídos seis números, correspondentes às seis direções, na ordem estabelecida por nós: estabeleçamos, por exemplo, a ordem Norte, Sul, Este, Oeste, Acima, Abaixo. Estes números aparecem contidos em um linha da matriz, ou array, de duas dimensões, PL (PL está por "plano").

Referindo-nos sempre à Figura 2, os seis números contidos em PL(1,1), PL(1,2), PL(1,3) PL(1,4), PL(1,5), PL(1,6) são os números dos lugares aos quais se chega deslocando-se desde o lugar 1 na direção indicada.

- Norte = 1
- Sul = 2
- Este = 3
- Oeste = 4
- Acima = 5
- Abaixo = 6

Se, desde o lugar 1 o jogador decide mover-se na direção 3 (para o Este), o elemento PL(lugar, direção), ou seja, PL(1,3) contém o número do novo lugar, neste caso 3 (moradia verde). Se colocamos LU igual a este número, teremos deslocado o personagem ao novo lugar.

O deslocamento nem sempre é possível. Para indicar a impossibilidade do movimento em uma certa direção desde um determinado lugar, colocamos um zero no elemento correspondente. Por exemplo, desde o lugar 1 não se pode ir para o oeste, porisso PL(1,4) contém um zero.

Resumindo: cada linha da matriz PL corresponde a um lugar, e cada coluna a uma direção. A cruz entre a linha e coluna indica

se é possível o deslocamento desde aquele lugar naquela direção e, caso afirmativo, o número do lugar ao qual se chega. Se LU é o lugar atual e D o número correspondente à direção escolhida, PL (LU,D) é o novo lugar se o movimento é possível, ou zero se não o é.

Voltemos a examinar o programa. A linha 220 espera a ordem do jogador e coloca D a zero (nenhuma direção introduzida). As linhas desde a 230 à 280 servem para colocar em D o código (número) da direção escrita, sem nenhuma pretensão (por hora) de empregar um método mais eficiente. Por exemplo, se o jogador escreveu "ESTE" a linha 250 executa um D = 3 (código de "Este"). Se a palavra introduzida não corresponde a nenhuma das direções possíveis, D permanece igual a zero.

A linha 290 comprova se foi introduzida uma direção válida. Em caso contrário (D vale zero), imprime "NÃO ENTENDO" e nos envia ao começo de loop.

A linha 300 consulta o plano contido na matriz PL para conhecer o lugar adjacente ao atual (LU) na direção indicada (D), e coloca em NL o número do novo lugar. Não o passa imediatamente a LU porque poderia ser zero (deslocamento impossível) e, neste caso, seria perdido o valor de LU.

Se o deslocamento não é válido, a linha 310 imprime "NÃO PODE" e volta ao começo do ciclo (LU não mudou); se o é, a linha 320 muda o valor de LU, deslocando assim ao personagem ao novo lugar.

Programa e dados

O conceito fundamental que está na base do visto é este: mudamos os dados, mas não mudamos o programa. Efetivamente, se quiser representar um plano completamente diferente, basta mudar o conteúdo da matriz de descrições DE$ e da matriz de conexões PL, mas o programa permanece igual: o trabalho foi feito uma só vez. Como vantagem secundária, mas não porisso menos importante, o programa ocupa muito menos espaço.

Um programa deste tipo é chamado "orientado por tabelas", já que seu comportamento depende exclusivamente dos dados contidos em uma tabela apropriada, neste caso as matrizes PL e DE$. Também é chamada "máquina de estados variáveis", enquanto que a diferença entre uma situação e outra (entre um lugar e o outro) depende exclusivamente do estado (valor) de uma ou mais variáveis; em nosso caso somente de LU (os puristas nos

perdoarão esta descrição não muito formal). O programa executa sempre no mesmo loop, onde quer que se encontre o aventureiro. Nos resta o problema de como introduzir os valores nas tabelas, ou seja, nas matrizes DES e PL.

Poderia ser feito desta maneira:

```
110 DE$(1) = "ESTA NA MORADIA VERMELHA"
120 DE$(2) = "ESTA NA MORADIA BRANCA"
130 DE$(3) = "ESTA NA MORADIA VERDE"
140 PL(1,1) = 0
150 PL(1,2) = 2
160 PL(1,3) = 3
...
```

Em seguida se vê que esta não é a solução mais prática: é desperdiçada muita memória para executar instruções repetitivas (pensem em um plano com uma centena de moradias). Existe uma técnica decididamente melhor, baseada em uma comodíssima instrução do BASIC que, por outro lado, falta em algumas linguagens "mais avançadas": a instrução DATA, que permite introduzir no programa uma lista de dados numéricos ou de cadeias. Portanto, podemos escrever nossas duas tabelas desta maneira:

```
300 DATA "Esta na moradia vermelha"
310 DATA 0,2,3,0,0,0
320 DATA "Esta na moradia branca"
330 DATA 1,0,3,0,0,0
340 DATA "Esta na moradia verde"
350 DATA 1,0,0,2,0,0
```

À descrição de cada lugar, seguem imediatamente os passos possíveis nas seis direções. Para ler estes dados utilizamos a instrução READ. Equivale a INPUT, mas lê os valores da lista de DATA em lugar de pelo teclado (a partir do ponto ao qual havia chegado a READ anterior).

Eis aqui ao que é reduzida a inicialização (preparação inicial) das variáveis para o programa que mostramos anteriormente:

```
110 NL=3:REM NUMERO LUGARES
120 DIM DE$(NL),PL(NL,6)
```

```
130 FOR LU=1 TO NL
140 READ DE$(LU)
150 FOR D=1 TO 6:READ PL(LU,D):NEXT D
160 NEXT LU
170 LU=1
```

Note que a linha 110 especifica o número de lugares, que é um valor fixo, ou seja uma constante. É uma boa regra indicar as constantes uma só vez ao princípio do programa, ao invés de espalhar por todos os lugares. Assim, as eventuais (na realidade seguras) modificações são muito mais simples e é menos fácil cometer erros. Seria ainda mais inteligente fazer ler o número de lugares dos DATA com um READ NL: o programa permaneceria inalterado também ao variar o número de lugares.

A linha 120 estabelece as dimensões de nossas tabelas, ou seja, as duas matrizes DE$ e PL. Em muitos BASICs não é necessário se as dimensões não superam o 10, como em nosso caso, mas se são indicadas tampouco importa. O loop aberto na linha 130 e fechado na 160 é repetido para cada lugar; três vezes. portanto no exemplo. A linha 140 lê a descrição do lugar no elemento correspondente da matriz de cadeia DE$. Enquanto que o loop na 150 lê o plano das direções na matriz bidimensional PL. Finalmente, a linha 160 estabelece que o personagem inicia o jogo no lugar 1.

CAPÍTULO VI

O COMPUTADOR ENTENDE PORTUGUÊS

Inteligência artificial (ou quase)

Muitos experts nos jogos de aventuras se perguntam com assombro com que diabos faz o computador para entender as frases escritas pelo jogador. Outros nem sequer o perguntam, como se fosse óbvio que o computador está dotado de uma determinada faculdade de compreensão. É ou não é uma máquina inteligente?

A resposta é: NÃO, o computador não é uma máquina inteligente. O computador é tonto como um asno, ainda que, ao contrário, é o soldado perfeito, o eterno sonho dos generais de todas as épocas: dotado de poderes notáveis obedece cegamente e sem emoções qualquer ordem, sem perguntar os motivos e sem prever as conseqüências. Faz tudo ao pé da letra, sem compreender minimamente o que está fazendo e sem perguntar se o significado literal das instruções recebidas corresponde realmente ao desejado pelo programador (caso bastante raro).

A inteligência não está no computador, mas no programa. Este último, é o momento de recordá-lo, somente é um seqüência de instruções escritas por um programador mais ou menos esperto. A inteligência que demonstra o computador é a de quem escreveu o programa. A vantagem do computador é que, com um pouco de técnica, é possível escrever um programa adequado para resolver não somente um problema específico, mas todos os problemas do mesmo tipo, economizando assim bastante trabalho. No capítulo anterior demos um exemplo desta possibilidade,

mostrando-lhes um programa que "entende" o conceito de direção.

Se o leram bem, terão claro que o programa não entende nada, mas que se limita a deslocar números e reproduzir escritos segundo regras bem definidas (a seqüência de instruções que forma o programa). A definição destas regras é, precisamente, tarefa do programador que quer dar à máquina uma aparência de comportamento inteligente.

O analisador sintático (parser)

Temos que escrever, portanto, um programa capaz de simular a compreensão dos conceitos que estão na base dos jogos de aventuras e que, fundamentalmente, se referm a

- plano, lugares e deslocamentos;
- objetos fixos e objetos transportáveis;
- compreensão e execução das ações do aventureiro.

O primeiro problema (plano e deslocamentos) já está resolvido. O segundo pode ser afrontado facilmente associando a cada objeto uma cadeia (sua descrição) e um número (o lugar no qual se encontra). Falaremos mais adiante dos detalhes. Decididamente, o problema mais complexo é o terceiro: entender a frase escrita no teclado e modificar, em conseqüência, o estado do jogo.

O programa que efetua a análise de uma frase (ou mais geralmente, de uma seqüência de símbolos) sobre a base de um conjunto de regras (ou sintaxe), é chamado parser ou analisador sintático. Um jogo de aventuras consiste, substancialmente, em um parser que, sobre a base das palavras introduzidas e de suas associações, decide o comportamento sucessivo do programa. Para escrever este programa é necessário, primeiro, definir as regras que governam o comportamento. Para fazer isto, tentaremos classificar as frases possíveis escritas pelo jogador:

- uma direção (deslocamento), por exemplo "NORTE";
- uma ação, expressa com uma só palavra, por exemplo "INVENTÁRIO";
- uma ação sobre um objeto, por exemplo "APERTA O VERDE".

Agora não entraremos em mais detalhes, deixando-os para

o capítulo dedicado às ações. Faremos notar somente que uma mesma ação pode ter efeitos distintos em lugares distintos (exemplo: "Lê o Cartaz"), que o objeto citado também pode estar ausente. temos que ter em conta esta e outras possibilidades se quisermos realizar um jogo com um bom nível de simulação.

Análise do léxico

Para facilitar o trabalho do analisador sintático é necessário separar as palavras que compõe a frase (sem tomar em conta eventuais espaços). Mas a manipulação das palavras como cadeias exige bastante espaço: é mais prático atribuir um número (código) a cada vocábulo e passar ao parser somente os códigos das palavras que compõem a frase. Também temos que descartar as palavras errôneas.

Estes trabalhos de menor importância são tarefas do analisador do léxico, a subrotina que recebe uma cadeia que contém a frase introduzida pelo jogador, a separa em vocábulos diferentes e restitui seus códigos numéricos. Às vezes, o analisador do léxico faz um trabalho que seria de competência do parser: reconhece e deixa de um lado os artigos, dado que não alteram o significado da frase e acrescentariam um trabalho inútil ao parser. Por exemplo, se o jogador escreveu:

"REPARA A ANTENA" a subrotina de análise do léxico separa as três palavras "REPARA", "A" e "ANTENA". O artigo "A" é descartado e são restituídos os códigos numéricos correspondentes a "REPARA" (27) e "ANTENA" (69). Será tarefa do parser decidir se este par de códigos corresponde a uma frase válida no lugar e na situação em que se encontra o aventureiro. Dado que nosso parser é bastante simples, as palavras seguintes à segunda (excluídos os artigos) são, simplesmente, ignoradas. Portanto não são possíveis frases como "Olha atrás do cartaz" e similares.

Intérprete e dados

Estamos quase preparados para começar; os detalhes podemos examiná-los durante o andamento. Desde o próximo capítulo se começa a escrever a aventura. Resumimos como funciona a técnica empregada para "A nave espacial condenada":

- o jogador introduz uma frase;

- o analisador léxico a separa em palavras e atribui um código a cada uma;
- o parser decide se os códigos (as palavras) indicam uma ação válida;
- se é assim, chama a subrotina BASIC que executa a ação correspondente.

Não citamos as direções, pois também estas, efetivamente, constituem uma ação como todas as demais, e não existe nenhuma necessidade de tratá-las diferentemente.

O programa final está composto por uma coleção de dados, por uma parte variáveis (as ações BASIC) e por outra parte fixos (analisador do léxico, parser e subrotinas em conexão). Esta parte fixa é perfeitamente reutilizável para outras aventuras, sem necessidade de modificações. Dado que interpreta uma série de dados proporcionados pelo programador, chamamos "intérprete" a esta seção reutilizável do programa. Também algumas das subrotinas BASIC fundamentais (as usadas para as direções, "Save", "Pega", etc.) são reutilizáveis tal qual ou com poucas modificações.

O programa é composto, portanto, de três partes:

- o intérprete, utilizável sem modificação para outras aventuras;
- os dados da aventura (vocábulos, plano, descrições, etc.);
- as subrotinas BASIC das ações (algumas delas reutilizáveis).

Uma boa notícia: para escrever uma aventura não é necessário compreender o funcionamento do intérprete, que constitui a parte decididamente mais complexa do programa. De todas as formas, para quem esteja interessado, na última parte do livro daremos uma detalhada descrição técnica de seu funcionamento.

CAPÍTULO VII

PLANO, DICIONÁRIO E OBJETOS

Nossa aventura

O objetivo deste livro é capacitá-los para escrever seu próprio jogo de aventuras. Referindo-nos ao programa da nave espacial condenada, recordamos que o intérprete, formado pelas linhas que vão até a 1160, pode ser reutilizado sem modificação (à parte do ajuste de algum GOSUB que chama às subrotinas).

A partir deste capítulo iniciamos a descrição das diferentes fases da construção de uma aventura, utilizando como exemplo a apresentada no segundo capítulo. Ao terminar, e para os que estejam interessados, descrevemos também o funcionamento do intérprete.

O plano

Uma vez estabelecida a disposição dos lugares nos quais é desenvolvida a aventura e suas conexões relativas não resta mais que introduzir estes dados no programa. Como dissemos no capítulo anterior, se trata somente de escrever uma lista de instruções DATA.

O plano dos lugares nos quais se desenrola "A nave espacial condenada" está representado na Figura 1, e acrescentamos números que identificam cada lugar. Existe 11 lugares numerados do 1 ao 11.

Para cada lugar é necessário proporcionar ao programa uma

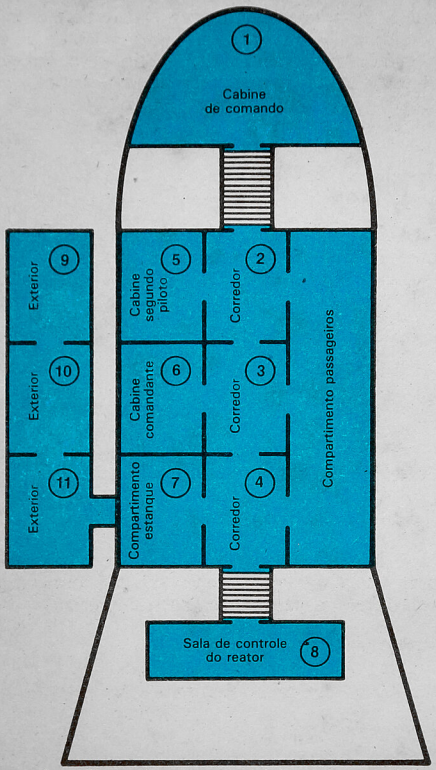

Figura 1. — Planta da nave espacial de passageiros "Neutronia".

descrição do mesmo e os números dos seis lugares contíguos (ver Capítulo V). Com referência à listagem (reproduzida no Capítulo II), as linhas desde a 4130 a 4230 contém estas informações, na ordem correspondente ao número do lugar (o lugar 1 na linha 4130, e assim sucessivamente).

Note que as descrições estão incompletas: para economizar memória omitimos o ponto final e o "Estas" inicial, que é acrescentado automaticamente pelo intérprete no momento de imprimir a descrição do lugar (se quer sabê-lo, linha 760).

Além disso, os números que representam os lugares adjacentes nas seis direções não estão escritos separadamente, mas recolhidos em uma só cadeia de doze caracteres (também para economizar memória). Os dois primeiros caracteres indicam o lugar adjacente ao Norte, os dois sucessivos o adjacente ao Sul e, em seguida, Este, Oeste, Acima e Abaixo.

Dado que é necessário indicar sempre dois caracteres, os números de uma só cifra vão precedidos de zero.

Ao término das linhas de DATA, o "FP" indica o final dos planos (linha 4230).

Em resumo, para introduzir um plano devem:

- desenhar cuidadosamente o esboço, com todas as conexões;
- numerar os lugares com números progressivos.
- para cada lugar escrever uma descrição e os números dos lugares adjacentes nas seis direções, na ordem N, S, E, O, A, B. Por exemplo:

  (Estas) no corredor, 2,4,0,6,0,0

  Recordemos que o zero indica uma direção não admitida. Depois voltam a escrever os seis números seguidos, estendendo-os a duas cifras se é necessário:

  02 04 00 06 00 00

  E, tirando os espaços, obteremos a cadeia:

  "020400060000"

  A descrição e esta cadeia, separadas por uma vírgula, formam a linha de DATA que utiliza o intérprete (confrontar com a linha 4150);

- por último, não esqueçam a cadeia "FP" que indica ao intérprete o final do plano. Podem escrevê-la em uma linha isolada de DATA ou, melhor , ao final do DATA do último lugar (linha 4230). Se esquecerem obterão erros estranhos

ao iniciar o programa (talvez em outra linha, quando o intérprete perde o fio da meada ao ler os DATA).

O dicionário

Uma vez definido o plano é necessário escrever todos os vocábulos que deverão ser "entendidos" pelo intérprete. Quanto mais rico fôr o vocabulário, mais fácil será simular um comportamento "inteligente" do jogo. "A nave espacial condenada" utiliza uns 50 vocábulos.

O trabalho é muito simplificado se é definido desde o princípio o maior número possível de vocábulos, acrescentando em seguida somente os referentes a variações ou possíveis melhoras do jogo. Começaremos, pois, por catalogar os termos necessários, associando a cada um, um número (à nossa escolha) que o intérprete utilizará em lugar do vocábulo, com uma grande economia de espaço.

Para uma melhor organização é muito cômodo subdividir as palavras em quatro grupos:

- Direções e ordens gerais, utilizáveis em outras aventuras (ex. "Norte", "Pega", "SAVE").
- Verbos (ex. "Abre", "Salta").
- Objetos transportáveis (ex. "Chave").
- Objetos fixos e de decoração (ex. "Armário").

A Figura 2 nos mostra os vocábulos usados em nossa aventura espacial, com seus respectivos códigos. Reservamos os códigos inferiores a 20 para os termos de uso geral (se não mudá-los ficará muito fácil voltar a utilizá-los em outros jogos). Depois subdividimos os restantes em três categorias, fazendo partir os códigos desde números arbitrários (20, 50 e 60). As cores (códigos 62, 63 e 64) servem para frases tipo "Pressiona o verde" ou "Pressiona o vermelho", e portanto são tratados como objetos fixos, de forma análoga a "Pressiona a tecla". Os artigos (que são ignorados) aparecem sob o mesmo código (7). Os sinônimos têm o mesmo código e estão escritos na mesma linha (ex. "O" e "Oeste").

Os códigos são sempre números arbitrários compreendidos entre 1 e 98, sem nenhuma implicação de ordem ou espaço ocupado. Uma subdivisão como a que mostramos sempre será cômoda para distinguir os distintos tipos de palavras.

Surge agora um problema técnico: a busca do dicionário é

A nave espacial condenada: dicionário

Código	Palavra	Código	Palavra	Código	Palavra
1	N,NORTE	2	S,SUL	3	E,ESTE
4	O,OESTE,W	5	A,ACIMA,SOBE	6	B,ABAIXO,BAIXA
7	O/,A/	8	PEGA	9	DEIXA
10	OLHA	11	SAVE	12	LOAD
13	COISA, INVENTÁRIO				
20	VESTE,PONHA	21	TIRA	22	ABRE
23	PUXA	24	EMPURRA	25	LÊ
26	APERTA	27	REPARA, AJUSTA		
50	CAPACETE	51	MACACÃO	52	TRAJE
53	SEGUNDO	54	CHAVE	55	MANUAL
60	CARTAZ	61	PULSADOR	62	VERDE
63	AMARELO	64	VERMELHO	65	ALAVANCA
66	INDICADOR	67	ARMÁRIO	68	CAMA
69	ANTENA	70	ETIQUETA		

Figura 2. — Vocábulos do dicionário na ordem correspondente a seus códigos.

muito lenta. Quando o analisador do léxico isolou uma palavra (ex. "ANTENA") e chama à subrotina que consulta o dicionário para encontrar seu código (o número que acabamos de atribuir), esta subrotina tem que observar todos os vocábulos até que encontre a palavra procurada. Se esta está ao final da lista, ou não existe, pode passar um tempo relativamente longo, que de todas as formas é enfadonho para quem joga.

A melhor solução é ter o dicionário em ordem alfabética, para assim poder usar uma subrotina binária de busca rápida (ver o livro n.º 9 da B.B.I.). Dado que a ordenação requer um certo tempo, é melhor inserir os vocábulos já ordenados por ordem alfabé-

Palavra	Código	Palavra	Código		
A	5	CARTAZ	60	OLHA	10
A/ (artigo)	7	CHAVE	54	PEGA	8
ABAIXO	6	COISA	13	PONHA	20
ABRE	22	DEIXA	9	PULSADOR	61
ACIMA	5	E	3	PUXA	23
AJUSTA	27		0	REPARA	27
ALAVANCA	65	EMPURRA	24	S	2
AMARELO	63	ESTE	3	SAVE	11
ANTENA	69	ETIQUETA	70	SEGUNDO	53
APERTA	26	INDICADOR	66	SOBE	5
ARMÁRIO	67	INVENTÁRIO	13	SUL	2
	0	LÊ	25	TECLA	61
B	6	LOAD	12	TIRA	21
BAIXA	6	MACACÃO	51	TRAJE	52
		MANUAL	55	VERDE	62
CAMA	68	N	1	VERMELHO	64
		NORTE	1	VESTE	20
CAPACETE	50	O	4	W	4
		O/ (artigo)	7		
		OESTE	4		

Figura 3. — Vocábulos do dicionário em ordem alfabética. (Os artigos **o** *e* **a** *deverão estar acompanhados pela barra (/) para diferenciá-los das abreviaturas de oeste e acima, respectivamente).*

tica. Isto quer dizer que o tempo economizado pelo jogador é pago com um pouco mais de trabalho por parte do programador (geralmente vale a pena, além de tudo se faz somente uma vez). Portanto, ao trabalho: o resultado final é apreciado na Figura 3.

Naturalmente, agora os sinônimos aparecem repartidos por diferentes pontos da lista. Quando ordenarem o dicionário comprovem bem: uma correta ordenação alfabética é fundamental. Em caso contrário, algum vocábulo poderia não ser reconhecido. Se tiverem dúvidas... consultem um dicionário.

Voltando ao programa, as linhas de DATA desde a 3990 a 4090 contém o dicionário, em ordem alfabética. Cada palavra vai

seguida por seu código. O "FD" final indica ao intérprete o final do dicionário.

Resumindo:

- cataloguem os vocábulos subdividindo-os em grupos;
- atribuam um código a cada vocábulo ou grupo de sinônimos;
- ordenem os vocábulos em ordem alfabética associando a cada um seu código;
- incluam o anterior no programa, em forma de DATA.

Os objetos

O aventureiro se encontra com objetos de muitos tipos. Alguns destes podem ser apanhados e transportados (como uma chave), outros simplesmente fazem parte da decoração (como uma cama). Todos podem ser mencionados ou manipulados (pelo menos com "Olha"). Os objetos transportáveis podem ser encontrados em vários lugares, ou em posse do aventureiro ou, provisoriamente, "fora do jogo".

O intérprete necessita uma lista dos objetos com suas distintas características. Os objetos são numerados, associando a cada um deles um número a partir de 1. Este número não tem nada que ver com o código associado no dicionário com o nome do objeto, mas serve para que o intérprete (e o programador) se refiram ao objeto em questão. A ordem é arbitrária, mas temos que ter em conta que os objetos são descritos pelo jogador (vejo...) por ordem de número.

A cada objeto é associada uma cadeia (sua descrição) e outros dois números: o código associado no dicionário à palavra que o descreve e o lugar no qual se encontra. A Figura 4 mostra a lista de objetos usados em "A nave espacial condenada".

O lugar contém outras informações:

- se o lugar é zero, o objeto é transportado pelo aventureiro;
- se o lugar é negativo, o objeto não pode ser apanhado (ex. a cama). Este é um típico truque para ter duas informações distintas (lugar e possibilidade de pegá-lo) no mesmo número;
- se o lugar é -99, o objeto está no "limbo", ou seja, fora do jogo.

A nave espacial condenada: objetos

Num.	Objeto	Código	Lugar
1	um indicador	66	−1
2	um pulsador	61	−1
3	uma etiqueta	70	−1
4	um macacão	51	1
5	um cartaz branco	60	−2
6	um cartaz amarelo	60	−4
7	uma cama	68	−5
8	um armário	67	−5
9	uma cama	68	−6
10	um armário	67	−6
11	um capacete	50	6
12	um pulsador vermelho	61	−7
13	um pulsador verde	61	−7
14	um indicador	66	−8
15	uma alavanca	65	−8
16	um pulsador vermelho	61	−8
17	um pulsador verde	61	−8
18	um cartaz vermelho	60	−8
19	um pulsador amarelo	61	−8
20	o segundo piloto	53	9
21	uma antena parabólica	69	−9
22	um traje	52	−99
23	uma chave	54	−99
24	um manual	55	−99

Figura 4. — Objetos existentes na aventura.

Por exemplo, o objeto número 6 está descrito como um cartaz no dicionário tem o código 60 ("CARTAZ"), é encontrado no lugar 4 (final corredor Sul) e não pode ser apanhado (o número do lugar é negativo).

Pode ter vários objetos com o mesmo nome (e portanto com

o mesmo código de dicionário), mas ninguém poderá pegar (ex. as camas 7 e 9, e as teclas, 2, 12, 13, 16, 17, 19) de forma que não podem nunca ser encontrados no mesmo lugar (o programa não se confunde freqüentemente, mas o jogador sim).

Na lista de DATA os objetos estão catalogados na mesma forma, na ordem de seu número e acabando com "FO" (final de objetos).

O número não está explícito na lista: o primeiro objeto catalogado tem o número 1, o segundo o número 2, e assim sucessivamente. As linhas desde a 4400 à 4510 contém a lista dos objetos utilizados na aventura (idêntica à da Figura 4).
Resumindo:

- catalogar os objetos utilizados e atribuir a cada um, um número progressivo;
- estabelecer a descrição de cada objeto (ex. uma "antena parabólica);
- escrever o código da palavra no dicionário (ex. o código de Antena, ou seja, 69);
- escrever o lugar no qual se encontra o objeto ao começo do jogo (ex. 9 para a antena), -99 se está fora do jogo ou zero se está transportada. Colocar o sinal menos se o objeto não pode ser apanhado (como no caso da antena, cujo lugar portanto é -9);
- introduzir tudo isto em linha de DATA.

É importante ter bem marcada a lista dos objetos com o correspondente número correlativo. De fato, o intérprete cria uma matriz LO% (Lugar Objetos) que contém o lugar de cada objeto, na ordem de seu número. O símbolo "%" (por cento) indica que se trata de uma matriz de números inteiros; somente serve para economizar memória e, em alguns BASIC, também tempo nos cálculos. Esta matriz LO% contém quase todo o estado da aventura se é executada a posição do aventureiro (contida na variável LU) e algumas informações acessórias. Todos os outros dados constantes (não mudam no curso do jogo). Atuando sobre LO% o programador pode mover os objetos ou transformá-los. Por exemplo, a instrução

```
LO% (23) = 7
```

leva o objeto 23 (a chave) ao lugar 7 (o compartimento estanque).

Dado que a descrição dos objetos visíveis é automática, a ins-

trução citada faz "aparecer" a chave, que pode ser apanhada, olhada, etc. Com a mesma técnica um objeto pode ser transformado, por exemplo fazendo desaparecer uma luz verde (colocando-a no lugar -99) e aparecer uma luz vermelha (colocando-a no lugar atual, LU ou melhor -LU, dado que se trata de um objeto que não pode ser apanhado). Do ponto de vista do jogador sempre é o mesmo objeto, mas o programador trabalha, na realidade, com objetos diferentes.

A manipulação do lugar dos objetos mediante LO% é o instrumento principal à disposição do programador.

Outros exemplos:

```
LO%(6) = 0
```

deixa o objeto 6 (o capacete) em posse do aventureiro, enquanto que:

```
LO%(24) = LU
```

coloca o objeto 24 (o manual) no lugar atual. Se estava em outro compartimento, "aparece"; se estava em posse do personagem é "deixado" no chão. Finalmente, a instrução:

```
IF LO%(23) = 0 THEN...
```

significa: se o objeto 23 (a chave) está no lugar zero (transportado), ou seja, se o aventureiro possui a chave...segue a conseqüência (por exemplo, o armário pode ser aberto).

CAPÍTULO VIII

AÇÃO!

Levar a cabo as ordens

Chegamos ao ponto mais delicado de todo o mecanismo: a execução das ordens dadas desde o teclado. É aqui onde se vê a "inteligência do programa (ou a estupidez); vejamos, portanto, o problema em todos seus detalhes. O jogador escreve uma frase composta por uma ou duas palavras significativas (excluídos, portanto, os artigos) que chamamos respectivamente, "a primeira palavra" (ou "verbo") e, se existe, "segunda palavra" (ou "nome"). O parser tem, sobre a base dos códigos destas palavras, que chamar a subrotina que executa a ação desejada.

Já deveria estar bastante claro o que significa executar uma ação: se trata de imprimir uma mensagem e/ou modificar o estado de algumas variáveis, tipicamente LU (o lugar do aventureiro) e LO% (o lugar de um ou mais objetos). Por exemplo, a frase "pega o manual" deve fazer partir uma subrotina BASIC que controla se o manual está presente (e portanto, se pode pegar) e, no caso afirmativo, desloca o mesmo manual ao lugar zero (transportado) e imprime a mensagem "Feito".

Mas não é em absoluto conveniente escrever subrotinas BASIC diferentes para "Pega o manual", "Pega o Macacão", "Pega o traje", etc. Uma só subrotina pode ser ocupada pelo verbo "Pega" em geral.

Esta é a técnica utilizada em muitos jogos de aventuras, técnica que também está ilustrada em vários livros sobre o tema: o analisador sintático chama a subrotina BASIC correspondente à

primeira palavra da frase (o verbo), deixando a esta subrotina a tarefa de distinguir entre os diferentes casos particulares. O "seletor" da subrotina a executar funciona mais ou menos desta forma:

```
500 IF C1=1 THEN GOSUB 1000
510 IF C1=2 THEN GOSUB 1010
520 IF C1=3 THEN GOSUB 1020
```

Se C1 é o código da primeira palavra da frase, o funcionamento está claro: a subrotina que começa na linha 1000 executa as ações referentes ao verbo com código 1 (por exemplo "Pega"), a 1010 as relativas ao verbo com código 2, e assim sucessivamente. Se os códigos partem de 1 e são correlativos, é utilizado um sistema muito mais simples e eficiente:

```
500 ON C1 GOSUB 1000,1010,1020,...
```

que tem exatamente o mesmo efeito. Se C1 vale 1 é executada a subrotina 1, ou seja, a primeira da lista (equivale a um GOSUB 1000), se vale 2, a segunda, e assim sucessivamente.

Problemas de compreensão

Como dizíamos, o sistema ilustrado é o mais utilizado. Provavelmente o motivo reside em sua simplicidade: o parser não tem que tomar decisões, mas deixar todo o trabalho às subrotinas que chama. Como acontece freqüentemente, se trata de um delineamento equivocado, ainda que à primeira vista pareça o contrário: a simplicidade do parser é paga com uma maior complicação das subrotinas, que são muitas, enquanto que o parser é um só, ou então, se assim é escolhido, com um comportamento "mais tonto" do programa. Consideremos, por exemplo, a frase "Pega o manual". Existem várias possibilidades:

- o manual está no mesmo lugar que o personagem,
- o manual se encontra em outro lugar ou fora de jogo,
- o manual já está em posse do aventureiro,
- o manual não é um objeto que possa ser pego.

Se consideramos estas possibilidades na subrotina de "Pega", deveremos fazer um trabalho análogo em muitas outras subrotinas ("Deixa", "Olha", etc.). Se não as consideramos, a simulação e o realismo são perdidos.

Habitualmente é utilizada uma solução deste tipo: se considera que a segunda palavra é um objeto e verifica-se que esteja presente. Mas também esta "solução" apresenta problemas. Consideremos a frase "Procura o manual": naturalmente o manual não

está presente, portanto o parser contesta: "Não entendo" (enquanto que desejaria poder imprimir uma frase apropriada, do tipo "Procura-o você, eu não tenho vontade"). Neste caso, o controle de presença é molesto. Uma situação deste estilo é apresentada, em "A nave espacial condenada", com os pressionadores de cor: a frase "Pega o verde" é executada ainda que não exista na moradia o objeto "Verde", mas somente o objeto "Pressionador". Por outra parte, a ação "Lê o manual" é executada somente se o manual está presente.

Conseqüentemente, muitas aventuras comerciais já entram em crise com frases relativamente simples, como "Olha a árvore" que produzem respostas, por exemplo, como "Não noto nada de particular" ainda que a árvore não esteja presente, ou, em outras situações similares, muitas contestam sistematicamente "Não entendo".

Poderíamos continuar com uma lista dos problemas, das soluções parciais e das correspondentes armadilhas e complicações, mas preferimos parar neste ponto e ilustrar uma solução mais flexível e que, à custa de uma ligeira complicação do parser (que o escrevemos de uma vez por todas), agora trabalho e problemas aos futuros programadores (ou seja, vocês).

Tabela das ações

O parser de "A nave espacial condenada" não é particularmente rápido, ainda que está acima da média dos contidos nas aventuras comerciais e dos apresentados em livros semelhantes.

Nosso parser funciona desta forma: o programa consulta uma tabela de ações que contém informações sobre as ações válidas, e em base a ela decide se executa uma subrotina BASIC, e qual. As subrotinas BASIC são numeradas a partir de 1 e são chamadas com uma instrução ON...GOSUB, como já vimos anteriormente. A tabela pode reconhecer quatro tipos de frases, compostas por:

1) Um par de palavras (verbo + nome) em um determinado lugar (exemplo "Aperta o pressionador" na sala de controle do reator).
2) Uma palavra específica (verbo) seguida de outra qualquer em um determinado lugar (exemplo, "Olha XXX em um compartimento com ilusões ópticas).
3) Um par de palavras (verbo + nome) em um lugar qualquer (exemplo "Lê o manual", em qualquer lugar).
4) Uma palavra específica (verbo) seguida de qualquer outra,

em um lugar qualquer (exemplo "Pega XXX", em qualquer lugar).

Em todos estes casos é possível especificar se o objeto indicado pela segunda palavra deve estar presente para que a subrotina BASIC seja executada, ou se a ação é válida de todas as formas também em ausência do objeto.

São reconhecidos também dois tipos de frases, compostas por uma só palavra, que na prática são como os tipos 1 e 2, mas com a segunda palavra ausente:

5) Uma palavra (verbo) em um determinado lugar (exemplo, "Este" no corredor).
6) Um palavra (verbo) em um lugar qualquer (exemplo, "Inventário").

Uma coisa muito importante: a busca na tabela tem lugar na ordem na qual catalogamos as diferentes possibilidades. Por exemplo: se o jogador escreveu "ESTE", o parser procura se está prevista a palavra "Este" no lugar atual do aventureiro (caso cinco). Se a encontra, executa a ação indicada. Se não a encontra, olha se está prevista a palavra "Este" como válida em qualquer lugar (caso seis). A encontra e chama a subrotina indicada (a de deslocamento). Se não a encontrasse, imprimiria "Não entendo". O mesmo acontece no caso de duas palavras: o parser olha primeiro se está previsto o caso um (esse verbo e esse nome nesse lugar), e somente se não o encontra passa aos casos 2, 3, 4 e finalmente imprime "Não entendo". A regra é que o caso particular tem preferência sobre o geral. Em outros termos: habitualmente a palavra "Este" tem determinado efeito (move para este), mas em um determinado lugar específico queremos que chame a uma subrotina em particular. Se não está tudo claro não se preocupe (contudo). Vamos ver algum exemplo que dissipe as dúvidas que podem ir surgindo.

Direção: Norte

Comecemos com o caso mais simples: as direções. Dissemos anteriormente que as direções não requerem um tratamento especial: são palavras como as outras, que trata somente de executar a subrotina adequada. Esta subrotina será uma só para todas as direções e decidirá sua atuação em base ao código da primeira (e única) palavra, cujo código está contido na variável C1 (com um pouco de imaginação adivinharão que C2 contém o código da

segunda palavra).

Portanto, a tabela das ações tem que conter estas informações:

- se a frase contém somente a palavra "N" ou um sinônimo (código 1), executa a subrotina BASIC número 1 (que se ocupa das direções);
- se a frase contém somente a palavra "S" ou um sinônimo (código 2), executa a subrotina BASIC número 1 (sempre a mesma);
- se a frase contém somente a palavra "E" ou um sinônimo (código 3), executa a subrotina BASIC número 1 (idem, como acima).

E assim sucessivamente para outras três direções. Portanto haverá seis informações na tabela das ações. Cada informação consiste em dois números: a síntese da frase e o número da subrotina a executar.

A síntese da frase é um número que se calcula da maioria seguinte:

LU*10000 + C1*100 + C2

Dito mais claramente: as primeiras duas cifras indicam o lugar no qual tem efeito a frase (zero se vale qualquer lugar), as duas seguintes são o código da primeira palavra e as duas últimas são o código da segunda palavra (zero se falta). Portanto, a síntese da frase "NORTE", válida em qualquer lugar, é construída assim;

Lugar:	00 (qualquer lugar, pode ser omitido)
Código 1:	01 (código de "NORTE" ou "N")
Código 2:	00 (nenhuma palavra)

O resultado ou síntese da frase (obtido colocando os três dados) é 000100. Dado que se trata de um número e não de uma cadeia, é igual a 100. Na tabela das ações, que o programa lê ao começo das linhas de DATA, é inútil desperdiçar espaço com zeros não significativos; porisso, para que seja executada a subrotina número 1 em resposta à frase "NORTE", basta escrever:

```
DATA 100,1
```

Do mesmo modo, a síntese de "SUL" é 200, a de "ESTE" 300,

a de "OESTE" 400, a de "ACIMA" 500 e a de "ABAIXO" 600. Dado que todas estas frases devem chamar à subrotina número 1, a tabela das ações conterá seis informações, cada uma composta de síntese de frase e número da subrotina a executar:

```
DATA 100,1,200,1,300,1,400,1,500,1,600,1
```

Olhemos a linha 4270 do programa: começa exatamente com estes dados. Para acelerar a busca das ações, a tabela é escrita em ordem correspondente ao número de síntese de frase. Agora não tem problemas, mas é importante não esquecê-lo.

Portanto, se o jogador escreve a frase "NORTE" no lugar 14, o processamento seguinte será:

- o analisador do léxico devolve os códigos das duas palavras: C1 = 1 (código de "Norte") e C2 = 0 (código de não tem palavra);
- o parser compõe a síntese de LU, C1 e C2, ou seja, o número do lugar atual, código da primeira palavra e código da segunda palavra. Obtém 140100 e procura este número na tabela das ações. Vamos supor que não o encontra: queria dizer que a ação "Norte" não está prevista no lugar 14;
- o parser compõe então a síntese de 01 e 02 somente, ou seja 100, e a procura na tabela das ações. A encontra: é uma ação prevista como válida em todos os lugares. Lê o número da ação a executar: 1;
- o parser chama à ação número 1 utilizando o seletor das ações (linha 1700), que cede o controle à subrotina da linha 1780, ou seja, à subrotina BASIC de direção (finalmente!).

Para não deixar mistérios sem solução, vamos ver o que ocorre na mal afamada subrotina que executa os deslocamentos (linhas 1780-1800). Ocorre exatamente o que já mostramos no capítulo 5, com a única diferença de que o número do novo lugar não é lido por uma matriz numérica mas por uma matriz de cadeia. DI$(LU) não é outra coisa mais que a cadeia de doze cifras que foi introduzida em seguida à descrição do lugar número LU (ver capítulo sete, plano, e linhas 4130-4230).

O MID$ da linha 1780 isola as duas cifras correspondentes à direção escolhida, baseando-se no código C1 (as primeiras duas para "Norte", as duas seguintes para "Sul", etc.), e o VAL da mesma linha as converte finalmente em um número; o número do novo lugar que, provisoriamente, está em A. Se a direção não é permitida, a linha 1790 o faz notar; de outro modo, a linha 1800 executa o deslocamento. Destaquemos que a linha 1780 somente funciona se os códigos do dicionário das seis direções são 1,2,3,4,5,6.

CAPÍTULO IX

ALGUMAS AÇÕES

As três mais simples

Na Figura 1 pode ser vistos nossos apontamentos relativos às ações gerais de "A nave espacial condenada". Pensamos propô-los sem retocá-los, ainda que contenham alguma imprecisão e nem sempre sejam perfeitamente coerentes, porque nos pareciam adequados para mostrar o trabalho paciente que tem entre bastidores de uma aventura, ainda que seja tão simples como esta.

Os apontamentos estão organizados desta maneira: na primeira coluna foram escritas as frases que têm que reconhecer o programa (reagrupadas se tem que executar a mesma subrotina, como as seis direções), na segunda coluna está a síntese numérica da frase (número lugar, código primeira palavra, código segunda palavra), segue o número da ação a executar (subrotina BASIC) e, finalmente, tem uma descrição concisa da mesma ação.

As ações da Figura 1 são as gerais, ou seja, as válidas em todos os lugares e independentes da aventura específica (salvo algumas exceções, que já veremos).

Comecemos com três ações simples: "SAVE", "LOAD" e "INVENTÁRIO" que, respectivamente, se ocupam de registrar a situação atual (muito aconselhada antes de empreender ações imprudentes), retomar uma situação anterior e catalogar os objetos que possui o aventureiro (com o cômodo sinônimo "COISA").

O mecanismo de execução destas ações funciona exatamente como o das direções, fazendo executar a adequada subrotina BA-

A nave espacial condenada: ações gerais

Frase	Síntese	Subrotina	Ação
S	0200		
E	0300		
O	0400		
A	0500		
B	0600	1	Move
PEGA XXX	0899	−2	se lugar (OB) = 0 "Já feito" se OB = não pode ser pego "Não é possível" se OB = Macacão e lugar (Traje) = 0 "Tira antes o Traje" se OB = Traje e lugar (Macacão) = 0 "Tira primeiro o macacão" se não lugar (OB) = 0 se OB = Capacete, Macacão ou Traje "Agora o tem colocado" se não "Feito"
DEIXA XXX	0999	3	se OB = 0 ou Lugar (OB) < > 0 "Não o tem" se LU > = 9 {fora lugar (OB) = −99 "Perdido no espaço" se não lugar (OB) = LU "Feito"
OLHA XXX	1099	−4	"Nada de particular"
SAVE	1100	5	guarda situação
LOAD	1200	6	recobra situação
COISA,INVENTÁRIO	1300	7	imprime inventário

Figura 1. — Primeiros apontamentos das ações gerais de "A nave espacial condenada".

SIC de ação (respectivamente 5, 6 e 7). Recordamos que os códigos das palavras são encontradas no dicionário preparado com anterioridade (capítulo 7).

As subrotinas BASIC de "Save" e "Load" são encontradas nas linhas 2000 e 2060 e, naturalmente, dependem do computador utilizado. São muito primitivas, no sentido de que somente está permitido o resgate de uma única situação, com nome fixo ("ASTRO", definido na linha 1010).

A subrotina de "Inventário" (linha 2120) faz uso, ao contrário, de uma subrotina do intérprete, a que descreve os objetos presen-

tes em um determinado lugar. Dado que os objetos possuídos pelo aventureiro estão no lugar zero, vale reutilizar a mesma subrotina. Convém deixar tal qual esta ação. Como inciso, o GOTO 590 ao final da linha 2120 é, conceitualmente, um GOSUB 590 seguido de um RETURN. O efeito é idêntico e é economizado tempo (na prática, a subrotina de inventário retorna utilizando o RETURN que está ao final da subrotina chamada).

Olha

A ação "OLHA XXX", ao contrário, é mais complicada: deve ser executada como resposta a uma frase composta pela palavra "OLHA" seguida de uma palavra válida qualquer e serve em um lugar qualquer. A síntese da frase é construída como antes:

Lugar: 00 (qualquer lugar, pode ser omitido)
Código 1: 10 (código de "OLHA")
Código 2: 99 (qualquer palavra válida)

Uma novidade: indicando 99 como código da segunda palavra é entendido "qualquer palavra válida, sempre que exista". A síntese da frase é, portanto, 1099 e, dado que temos que executar a ação 4, na tabela deveremos escrever:

DATA 1099,4

Ao contrário, na tabela é encontrada:

DATA 1099,-4

(mais ou menos pela a metade da linha 4290). Esta é uma comodidade mais: se o número da subrotina é negativo, significa que o parser tem que controlar que a segunda palavra se refira a um objeto presente ou transportado, isto é, situado no lugar atual ou em posse do aventureiro. Em caso contrário, imprime "Aqui não está" e se nega a chamar a subrotina indicada.

Dado que para olhar um objeto este último tem que estar presente, vamos colocar o sinal menos e nos asseguramos assim que a subrotina número 4 seja executada somente se o jogador olha a um objeto realmente visível.

Como podem ver na linha 1970, a subrotina número 4 é limitada a imprimir "Não noto nada de particular". De fato, se trata da

subrotina geral de "Olha...", que está ao final da lista (caso 4 dos enumerados no capítulo 8) e serve para dar, de todas as formas, uma resposta, em ausência de ações mais interessantes (como "olha o indicador"). Na gíria técnica é uma subrotina de "default", ou seja, a executar em ausência de outras subrotinas com preferência mais alta.

Pega

Chegamos agora a duas ações fundamentais nos jogos de aventuras: pegar e deixar os objetos. Comecemos pela primeira.

Se quiser um mínimo de realismo, na ação "PEGA" é necessário considerar que o objeto citado:

- tem que estar presente;
- não pode estar já em posse do aventureiro;
- tem que poder ser pego.

A primeira condição é automática: basta colocar um sinal menos diante do número da ação (ver Figura 1: o número da ação a executar é -2). Para satisfazer as outras duas, a subrotina chamada é delineada mais ou menos desta maneira:

- se o objeto já está em poder do aventureiro imprimir "Já feito" e voltar;
- se o objeto é do tipo dos quais não podem ser pegos (lugar negativo, ver capítulo 7), imprimir "Não é possível" e voltar.
- se tudo é correto, deslocar o objeto ao lugar zero (transportado), imprimir "Feito" e voltar.

Para referir-se ao objeto citado na frase, é utilizada a variável OB, na qual o intérprete coloca o número do objeto (não o código do dicionário: ver capítulo 7) citado como segunda palavra.

Atenção: se o objeto não está presente no lugar LO ou transportado, OB contém zero e não é utilizado. Portanto, LO%(OB) é o lugar do objeto citado como segunda palavra. Nós temos o costume de levar nossos apontamentos em uma forma de "pseudo-código", ou seja, em uma espécie de linguagem estruturada da qual dou, em seguida, um exemplo reescrevendo parte da subrotina de "Pega":

```
se lugar (OB) = 0
```

```
        "Já feito"
se OB = não pode ser pego
        "Não é possível"
se no lugar (OB) = 0
        "Feito"
```

Não existe muita diferença em relação ao anterior; é somente mais conciso. As palavras "se" e "se não" ou "de outro modo" podem ser substituídas pelas equivalentes em inglês "if" e "else"; a vantagem está na brevidade e na semelhança com instruções BASIC, o que simplifica o trabalho de tradução ao programa. Além disso, escrevemos simplesmente entre aspas um resumo das mensagens a imprimir e fazemos uso do escalonamento, que consistem em deslocar para a direita (escalar) as operações (instruções) relativas a um determinado if, para vê-las mais claramente. Dado que este livro não é um curso de programação estruturada, pararemos aqui. De todas as formas, pensamos que nossas anotações são bastante legíveis.

Voltando à subrotina de "Pega", pode ser traduzida ao BASIC desta maneira:

```
1830 IF LO%(OB) = 0 THEN PRINT "JA FEITO":RETURN
1840 IF LO%(OB) =  0 THEN PRINT "NAO E POSSIVEL":RETURN
1870 LO%(OB) = 0
1890 PRINT "FEITO":RETURN
```

Note como a subrotina corresponde exatamente aos apontamentos. Em particular, a linha 1840 verifica que o objeto citado pode ser pego comprovando que seu lugar não seja negativo (se é negativo, significa "que não pode ser pego"; ver capítulo 7).

A subrotina do "Pega" utilizada em "A nave espacial condenada" é ligeiramente mais complicada, porque tem em conta alguns casos particulares: impede colocar o macacão se já colocou o traje (e vice-versa), e imprime "Agora o tens posto" em vez de "Feito" se o personagem pega o capacete, o macacão e o traje. Verifiquem estas diferenças entre os apontamentos da Figura 1 e as linhas 1830-1890 (na prática, foram acrescentadas as linhas 1850, 1860 e 1880). Em uma de suas próprias aventuras basta tirar estas linhas acrescentadas para ter uma subrotina geral do "Pega".

Passamos à ação recíproca: a de deixar um objeto. Aqui o trabalho é mais simples: se o objeto não é transportado não pode ser deixado. Dado que queremos imprimir "Não o tem" se o objeto não é transportado (esteja presente ou não), tem que evitar o controle preliminar do intérprete (que imprimiria "aqui não está" em caso de ausência). Para evitar o controle basta utilizar um número de ação positivo (efetivamente, é 3, e não -3).

Mas existe uma armadilha: Chegados a este ponto não basta olhar L0%(OB) para saber onde está o objeto. Efetivamente, como dissemos antes, se o objeto não está presente ou transportado OB vale zero e L0%(OB) produzirá, teoricamente, um erro. Na realidade, L0%(OB) contém zero (este elemento não foi utilizado nunca), mas isto não significa em absoluto que o objeto esteja transportado, significa somente que estamos olhando uma variável sem significado; em suma, temos que comprovar que OB não vale zero, porque senão quer dizer que o objeto, certamente, não pode ser deixado. Portanto, a ação de "Deixa" é:

```
se OB = 0 ou lugar (OB)<>0
    "Não o tem"
Se não
    lugar (OB) = LU
    "Feito"
```

Assim como para a ação "Pega", também para "Deixa" existe um caso particular que concerne à "A nave espacial condenada": se o número do lugar é maior ou igual a 9 (ou seja, 9, 10, 11) nos encontramos no espaço exterior e qualquer objeto deixado será perdido, transferindo-o ao lugar -99 (o "limbo"). As linhas de programa desde a 1920 até a 1940 são a subrotina de "Deixa".

Como em todos os programas não existe uma única forma de resolver os problemas. Se tivesse sido possível escrever uma subrotina mais "limpa" para "Deixa" e prever a frase "Deixa XXX" nos lugares 9, 10, 11. Nossa solução é mais curta, mas talvez menos aconselhável para "PEGA O CAPACETE", "PEGA O MACACÃO" e "PEGA O TRAJE", onde, ao querer destacar as três ações espaciais, as coisas ficavam mais complexas. Era necessário, na prática, repetir quatro vezes os controles da subrotina de "Pega". Provavelmente não valia a pena.

CAPÍTULO X

A BORDO DO "NEUTRONIA"

Ações sob medida

As ações que vimos até agora são de caráter geral, reutilizáveis para outras aventuras. Agora se trata de escrever ações específicas para uma determinada aventura, precisamente a de "A nave espacial condenada". Já deveriam ter muito claro que, à diferença do que ocorre em muitos outros programas para aventuras, estas ações específicas funcionam exatamente igual àquelas gerais, graças à tabela das ações. Também a subdivisão que fizemos nos apontamentos é simplesmente uma questão de comodidade.

Não descreveremos detalhadamente todas as ações; nos limitaremos a comentar as mais significativas, deixando-lhes o útil exercício de examinar as outras e entender como funcionam. Iniciamos em seguida com as ações válidas em todos os lugares da nave espacial, que podem ver resumidas na Figura 1.

Capacete, macacão e traje

Já as três primeiras ações ("Veste Capacete", "Veste Macacão", e "Veste Traje", onde omitimos os artigos como faz o programa) merecem nossa atenção. Para simplificar o jogo estabelecemos que pegar um destes três objetos equivale a vestí-lo (veja a subrotina de "Pega" do capítulo anterior). Então parecia conveniente indicar simplesmente "Veste" como sinônimo de "Pega" (mesmo código no dicionário); deste modo, as frases "Pega Macacão" e "Veste

A nave espacial condenada: ações válidas em qualquer lugar

Frase	Síntese	Subrotina	Ação
Veste Capacete	2050	−2	goto Pega XXX
Veste Macacão	2051	−2	goto Pega XXX
Veste Traje	2052	−2	goto Pega XXX
Deixa Capacete Deixa Macacão Tira Capacete Tira Macacão	0950 0951 2150 2151	−10	if lugar(OB) 0 "Não o tem" if LU = 9 ou (LU = 7 & Var2 = 1) fora "Aaaagh" Morto else goto Deixa XXX
Tira Traje	2152	−3	goto Deixa XXX
Olha Macacão	1051	−11	"É seu macacão para ativ.extraveic."
Olha Segundo	1053	−12	"Tem o capacete deteriorado"
Olha Cartaz Olha Manual	1060 1055	−13	"Não é melhor ler?"
Olha Traje	1052	−14	"Pesado, tratado com chumbo"
Lê Manual Abre Manual	2555 2255	−15	if lugar(OB) = LU "Pegue-o com a mão antes" else "Instruções: para ativar..."
Repara Antena	2769	−29	"Necessitamos o segundo"

Figura 1. — Ações de "A nave espacial condenada" válidas em qualquer lugar.

Macacão" teriam exatamente o mesmo efeito. Lamentavelmente, nem sempre a simplicidade é boa: com este sistema seriam aceitas frases como "Veste Manual" ou "Veste Chave".

Com relação a isto existe dois pontos de vista: o primeiro (típico de muitos autores) é o de "A quem importa? Ninguém dirá nunca uma frase assim". Ao contrário, pensamos que o realismo e o interesse de uma aventura são medidos também, e muito, pelo cuidado nos detalhes. Por sorte, o parser e a tabela de ações tornam simples a adição de novas possibilidades, e assim também os preguiçosos poderão construir uma aventura rica em detalhes sem muito esforço.

Por exemplo, é muito fácil fazer que "Veste Capacete" tenha o mesmo efeito que "Pega Capacete". É eficiente colocar na tabela de ações as sínteses de "Veste Capacete":

Lugar:	00 (em qualquer lugar)
Veste:	20
Capacete:	50

Portanto, a síntese é 2050, número que deve ser inserido em seu lugar na tabela (que, recordamos, vai em ordem de número de síntese). Em seguida é colocado o número da ação (subrotina BASIC) a executar. Dado que a frase tem que ter o mesmo efeito que "Pega Capacete", basta colocar o mesmo número que a subrotina de "Pega", ou seja, -2 (o sinal menos, como sempre, quer dizer que o objeto tem que estar presente). Na Figura 1 a indicamos como "goto Pega XXX" para recordar que esta ação tem o mesmo efeito que a outra. Isto mesmo serve, obviamente, para "Veste Macacão" e "Veste Traje".

Agora, portanto, a frase "Veste Capacete" tem o mesmo efeito que "Pega Capacete", mas a frase "Veste Chave" não é reconhecida em nenhuma parte e produz assim a resposta "Não entendo". Em uma aventura bem feita também deveria ser prevista uma resposta standard por default para o caso "Veste XXX", ou seja, para todas as tentativas de serem colocados objetos que não são adequados (por exemplo: "Não sabia que você podia mudar de forma. Pode transformar-se em uma larva?"). Melhor ainda, deveria ser prevista uma resposta adequada para cada verbo quando não fosse usado no modo ou com o objeto correto. Senão, a frase "Não entendo" seguidamente seria pesada: a variedade mantém o interesse do jogador. Com referência a Capacete e Macacão observem (na Figura 1) as frases "Deixa Capacete", "Tira Capacete", etc.

Dado que seria inverossímil permitir que um astronauta tirasse o capacete e o macacão, sem sofrer danos enquanto se encontra no vácuo, temos que considerar esta possibilidade: se o personagem realiza a ação no exterior da nave espacial (lugar > = 9), ou no compartimento estanque (lugar 7) quando este está aberto ao vácuo (variável 2, como veremos mais tarde), acontece o inevitável. Se não é essa a situação é executada a subrotina normal "Deixa XXX". Para fazer é suficiente executar um GOTO em BASIC à subrotina em questão, como pode ser visto na linha 2220 do programa. As linhas 2210-2230 constituem a subrotina de ação.

Ações particulares e gerais

Ainda que nesta aventura não existia exemplos adequados, recordamos que uma mesma ação pode ter um efeito distinto em diferentes lugares. Por exemplo, a frase "Pede ajuda" somente funcionaria próxima da jaula dos leões (imaginem vocês o efeito), enquanto que em outros lugares seria contestada com um desolador "Ninguém está próximo para ouví-lo".

É certo que neste caso pode ser escrita uma subrotina válida para todos os lugares (cuja síntese iniciará, portanto, com 00) e efetuar uma comparação (IF LU = 47 THEN...) ao princípio da subrotina para estabelecer se nos encontramos em um lugar especial, mas é melhor utilizar duas subrotinas distintas: uma geral (válida em todos os lugares), que dá uma resposta por default, e uma específica (válida em um só lugar) que realiza o efeito desejado. É o parser que se ocupa de dar preferência a esta última se o aventureiro se encontra no lugar indicado. Recordamos que a ordem de preferência das distintas ações já foi descrita no capítulo 8.

Ações específicas

A Figura 2 nos mostra a lista das ações que somente têm efeito se a frase é pronunciada em um determinado lugar. Se a frase é enunciada em lugar distinto ao indicado, o parser não a encontrará; então olhará se a tabela contém a mesma ação válida em todos os lugares. Se não aparece, imprimirá "Não entendo".

A nave espacial condenada: ações válidas em um lugar específico

Cabine (1):			
Olha indicador	011066	16	"Temperatura"
Lê etiqueta	012570	17	"SOS Galático, somente para emergências"
Aperta pulsador	012661	18	"Antena exterior defeituosa"
Corredor Norte (2):			
Lê Cartaz	022560	19	"Escada: entrada reservada..."
A	020500	20	LU = 1 cabine if lugar (Segundo) = 9 fora "Se estivesse aqui..."
E	020300	21	"Melhor não acordar os passageiros"

Figura 2. — Ações de "A nave espacial condenada" válidas somente em um lugar.

Corredor (3):			
E	030300	21	"Melhor não acordar os passageiros"
Corredor Sul (4):			
Lê Cartaz	042560	22	"O: Atenção: mor. despressurizada.."
E	040300	21	"Melhor não acordar os passageiros"
Cabine Segundo (5):			
Abre Armário	052267	23	if lugar (Manual) −99 "Já feito" else if lugar (Chave) 0 "Fechado com chave" else "Feito" Lugar(Manual) = LU Lugar(Traje) = LU
Cabine Comandante (6):			
Abre Armário	062267	24	"Vazio"
Compartimento Estanque (7):			
E	070300	37	if var2 = 0 fechado LU = 4 corredor else goto 1 (Direções)
O	070400	38	if var2 = 1 aberto LU = 11 exterior LU = 11 exterior else goto 1 (Direções)
Aperta pulsador	072661	25	"Aperta Vermelho ou Aperta Verde"
Aperta Vermelho	072664	26	if var2 = 1 aberto "Click" else "Se abre.." var2 = 1 if lugar(Macacão) <> 0 lugar (Capacete) <> 0 "Aaaagh" Morto if objetos móveis em LU lugar(objetos) = −99 "Perdidos no espaço"
Aperta Verde	072662	27	if var2 = 0 fechado "Click" else "Se fecha..." var2 = 0 if lugar(Segundo) = 0 and lugar(Chave) = −99 "Revive" lugar(Chave) = LU

Figura 2. — Continuação.

Sala controle reator (8):			
Lê Cartaz	082560	28	"Emergência a vista: Traz..."
Olha Indicador	081066	16	"Temperatura"
Aperta Pulsador	082661	30	"Aperta Vermelho, Amarelo ou Verde"
Aperta Vermelho	082664	31	"Click" "Tubulação que perde" if var1 = 0 var1 = 1 else var1 = 0 timer1 = timer1 div 2
Aperta Amarelo	082663	32	"Click" if var1 = 1 var1 = 2 if lugar(Traje) 0 "Você sente regular" else var1 = 0 timer1 = timer1 div 2
Aperta Verde	082662	33	"Click" if var1 = 2 var1 = 3 if lugar(Traje) 0 "Excesso de radiações" Morto else var1 = 0 timer1 = timer1 div 2
Empurra Alavanca	082465	34	"Clank" if var1 = 3 Final. Felicidades! else var1 = 0 timer1 = timer1 div 2
Puxa alavanca	082365	35	"Clunk" var1 = 0 timer = timer1 div 2
Exterior (9):			
Olha Antena	091069	36	"Parece avariada"

Figura 2. — Final.

Note que os números das ações (subrotinas BASIC) a executar são todos positivos: dado que em nenhuma destas ações a segunda palavra indica um objeto móvel (poderia acontecer em outra aventura) e dado que o lugar é conhecido (a síntese da frase começa com o número do lugar no qual tem que ter efeito a ação), não existe nenhuma necessidade de comprovar a presença do objeto citado (por exemplo, o armário).

A frase "Repara Antena" foi colocada entre as que são válidas em qualquer lugar (Fig. 1), porque tem que ser contestado "Aqui não está" automaticamente, por efeito do sinal menos, fazendo intuir assim que a antena tem que estar em alguma parte (muitos jogadores escrevem esta frase na cabine de comando). A primeira frase interessante é "A" ("Acima") no lugar 2 (corredor). Sua síntese está feita da forma habitual, mas indicando um número de lugar:

Lugar: 02
Primeira palavra: 05
Segunda palavra: 00

Portanto, a síntese é 020500, ou seja 20500. Dado que uma ação local tem preferência sobre uma geral, quando o jogador escreve "A" no lugar 2 não é executada a subrotina geral de movimento (subrotina 1), mas a específica assinalada neste caso (subrotina 20). Recordamos uma vez mais que as ações da tabela estão na ordem do número de síntese.

A subrotina que executa a ação (2800-2830) não somente é encarregada de transferir ao aventureiro à cabina de comando (LU = 1), como também imprime uma mensagem se o segundo piloto contudo não foi salvo. Esta é a técnica normal para fazer com que aconteça algo durante a transferência desde um lugar a outro.

Uma subrotina parecida é a número 21, que imprime uma mensagem ("Melhor não acordar aos passageiros..."), porém não permite o deslocamento (simplesmente, não muda LU).

Na ação "Abre armário", no lugar 5 (cabine do segundo) é comprovado que o manual não apareceu antes. Somente se ainda está no "limbo" (lugar —99), é comprovado se o personagem possui a chave. E unicamente neste caso aparecem o traje e o manual. A subrotina BASIC está nas linhas 2960-2980; a partir de agora não indicaremos mais linhas. Podem procurá-las vocês lendo as REM, ou então as instruções ON... GOTO...

O compartimento estanque

O compartimento estanque é um lugar interessante, ainda que não coloque muitos problemas aos jogadores amantes da ciência de ficção. Dado que pode estar em dois estados, isto é, aberto para o corredor ou para o espaço exterior, é necessária uma va-

riável para saber em que estado se encontra. A variável BASIC V2 (se, existe uma V1). Se vale zero o compartimento estanque está fechado (com relação ao espaço exterior), se vale 1 está aberto. O pressionador vermelho o abre e o verde o fecha.

Seria absolutamente errado, e não somente inútil, indicar um número de subrotina negativo para as ações "Aperta vermelho" e "Aperte verde". De fato, o parser procuraria os códigos correspondentes no dicionário às palavras "Vermelho" e "Verde", que não existen ; existem somente dois objetos com código "Pressionador" e são precisamente os números 12 e 13 (ver capítulo 7, Figura 4).

Ao abrir o compartimento estanque, as forças do ar no vácuo leva todos os objetos que, eventualmente, foram deixados no chão. Por um simples favor dos autores geralmente isto não acontece. Teria sido uma faina fazer a aventura de forma que tivesse algum objeto essencial pelo chão no momento da abertura do compartimento estanque. A propósito, é óbvio que não é possível sobreviver no vácuo sem macacão e capacete.

Nos damos conta, ao escrever estas linhas, de ter passado por alto uma possibilidade: se o aventureiro deixa no chão o segundo piloto antes de fechar o compartimento estanque, este não revive. Na resposta à frase "Aperta o verde", tem que prever um "if lugar (Segundo) = 0 ou lugar (Segundo) = LÚ...". Nenhum dos revisores de programa ficou atrapalhado aqui, mas é melhor prevê-lo porque senão a solução da aventura pode ser impossível.

O reator

A sala de controle do reator é, provavelmente, o lugar da nave espacial onde é possível realizar o maior número de ações. Como vocês sabem, a ação final da aventura consiste em desativar o reator com base às informações conseguidas de um manual incompleto.

Naturalmente, terão compreendido que para desativar o reator tem que executar exatamente ao contrário à seqüência de colocar em funcionamento o mesmo. Dado que para este último temos que puxar a alavanca, apertar o pressionador verde, apertar o pressionador amarelo e, por último, apertar o pressionador vermelho, a seqüência oposta será: apertar o pressionador vermelho, o amarelo, o verde e, obviamente, empurrar a alavanca.

Parece que os aventureiros não muito espertos têm dificuldades para imaginar uma alavanca típica de quadro de controle

do tipo das usadas para manejar os guindastes e as escavadeiras.

Voltaremos sobre o tema da dificuldade quando falarmos da revisão e da colocação em ponto. Por hora, nos interessa explicar o mecanismo de funcionamento do quadro de comandos:

- se uma ordem foi dada na seqüência correta, aumenta um contador;
- se uma ordem foi dada fora da seqüência correta, coloca a zero o contador e reduz à metade o tempo disponível (reaquecimento por manobra errada).

Do fator tempo falaremos no próximo capítulo: agora diremos que a variável T1 (timer-temporizador-1) é diminuída automaticamente em cada ciclo; quando chega a zero tudo explodirá silenciosamente (no vácuo). Dividindo T1 por 2, se reduz à metade o tempo disponível. O "div" na expressão "timer 1 = timer 1 "div" 2" significa divisão inteira, ou seja, sem resto (Como se faz em BASIC? Olhem a linha 3830).

Voltando ao mecanismo de controle do reator, a variável V1 (var 1 nos apontamentos) é o contador de desativação, que ao começo do jogo vale zero. A ação "Aperta Vermelho" (a primeira da seqüência) coloca V1 = 1 somente se anteriormente valia zero, ou seja, se não tinha sido executada nenhuma outra manobra (ou se uma manobra equivocada colocou tudo a zero); a ação "Aperta Amarelo" (a segunda) coloca V1 = 2 somente se valia 1, ou seja, se havia apertado o vermelho e somente o vermelho; "Aperta verde" coloca V1 = 3 somente se valia 2, ou seja, depois da seqüência vermelho-amarelo; finalmente, "Empurra Alavanca" acaba o jogo somente se V1 valia 3, ou seja, somente depois da seqüência correta vermelho-amarelo-verde. Cada manobra fora da seqüência envia à linha 3830 do programa, que parte pela metade o temporizador T1 e coloca a zero V1, obrigando a começar a seqüência desde o princípio.

A propósito: o traje serve somente para impedir que um aventureiro mal-intencionado consiga, ao observar no indicador de temperatura o fato das diferentes manobras (não o haviam pensado?), desativar o reator sem ter salvo primeiro ao segundo piloto: a penúltima manobra ("Aperta Verde"), faz com que o personagem morra por excesso de radiação (assim aprende a não passar rápido).

CAPÍTULO XI

ÚLTIMOS DETALHES

Uma vez definido o guia preparem o dicionário, o plano, a lista de objetos, a tabela das ações e as subrotinas que as levam à prática; estamos quase preparados para a "montagem" final da aventura, na forma de um programa em BASIC. Somente faltam as subrotinas de introdução, de final de jogo (para bem ou para mal), e de passagem do tempo. Vamos vê-las.

Prolegômenos

A introdução é encontrada no programa entre as linhas 1540-1670. Naturalmente, pode ter um número diferente a linha inicial, mas temos que recordar-nos então de modificar a chamada do intérprete na linha 720. Em primeiro lugar imprimam os títulos e uma mensagem introdutória à aventura, depois inicializem as variáveis que o intérprete deva conhecer; em nosso caso, o lugar inicial e as variáveis especiais utilizadas por esta aventura (T1, V1 e V2, ou seja, o timer, o contador do reator e o estado do compartimento estanque).

Precisamente a linha 1670 faz este trabalho; LU = 6 significa que o aventureiro começa o jogo no lugar 6 (a cabine do comandante), T1 = 100 quer dizer que a aventura é resolvida em 100 turnos (salvo manobras erradas, que reduzem o tempo disponível). V1 = 0 e V2 = 0 são instruções supérfluas, dado que em BASIC

todas as variáveis numéricas contém zero ao princípio do programa, mas não custa muito indicá-lo de maneira explícita. Na subrotina de "Save", temos que recordar-nos de gravar em disco ou em cassete não somente LU e LO%, mas também estas variáveis: T1, V1 e V2. O mesmo vale para a subrotina de "Load" (ver linhas 2000 e 2060, a sintaxe depende do computador utilizado).

Finais

O final positivo (aventura resolvida) não apresenta problemas: basta imprimir a mensagem de em boa hora adequado e terminar o programa com a instrução END. Isto pode ser feito diretamente na subrotina que executa a ação final (em nosso caso, "Empurra Alavanca"), como pode ser visto nas linhas 3640-3770. O GOSUB 1120 da linha 3650 serve para evitar que uma série de mensagens muito longas desapareça da tela antes de que o jogador tenha tido tempo de ler a mesma. Usem-na todas as vezes que acreditam oportuno.

O final negativo (morto) também imprime uma mensagem adequada (linhas 1200-1320), depois pergunta se quer voltar a começar desde o princípio. Em caso afirmativo, um simples RUN faz iniciar o programa, senão um END o termina. As linhas 1340-1360 são reutilizáveis tal e como aqui aparecem. Note que a 1340 extrai o primeiro caracter da resposta, de forma que aceite "S", "SIM", "N", "NÃO"; não são aceitos outros caracteres iniciais, pois voltaria a propor a pergunta.

A subrotina de "morto" somente é chamada por outras subrotinas do programa (exemplo: linha 3130), não pelo intérprete. Portanto, não existe nenhum problema em trocar os números da linha, somente temos que tê-los em conta nas correspondentes chamadas. Além disso, dado que a subrotina acaba com "END" ou com "RUN", pode ser chamada tranqüilamente com um GOTO desde qualquer nível de subrotina (nao existe problemas de RETURN).

Temporização

Vejamos finalmente o mecanismo de passagem do tempo, que encontrarão nas linhas 1400-1500. No caso de modificar os números de linhas, é necessário mudar também a chamada à linha 780 do parser.

A temporização está realizada sob medida para esta aventura, mas o princípio é válido para todos os casos nos quais existe passagem do tempo (por exemplo, uma vela que se consome). O parser chama a subrotina de tempo cada vez que o jogador introduz uma frase. A subrotina diminui o tempo restante (linha 1400) e depois verifica se é o momento em que deve ocorrer algum dos acontecimentos predispostos, que neste caso são distintas mensagens a imprimir (linhas 1410-1430) e a mensagem final de destruição quando o tempo venceu definitivamente. À parte deste último caso, a linha 1440 normalmente volta ao parser.

Montagem

Agora que estão todos os elementos definidos, pode ser procedida a construção do programa completo. Sugerimos um procedimento para a montagem de suas aventuras. Está claro que é conveniente usar de vez em quando o "SAVE", senão lhes faltará seguramente a corrente, devido à primeira lei de Murphy ("Se algo pode ir mal, irá mal"). Pelo mesmo motivo não empreguem o mesmo nome, mas nomes e números correlativos (exemplo: CAVERNAS1, CAVERNAS2, etc.). Se dispõe de cassete gravem alternativamente em todos os dois lados. Cada cinco ou seis gravações façam uma cópia em outro disco ou cassete.

Estão preparados? Pois adiante:

1) Carreguem o intérprete na memória, ou seja, as linhas desde a 100 à 1160.
2) Introduzam as linhas com os DATA do dicionário, comprovando uma vez mais que os vocábulos estejam em ordem alfabética, seguidos cada um de seu próprio código, e que não falte o "FD" ao final. Convém utilizar um número de linha bastante alto, digamos 8000, para não obstruir depois a escrita das ações.
3) Introduzir as linhas com os DATA do plano, verificando que estejam na ordem correta (a partir do lugar 1), que cada descrição esteja seguida da lista dos passos e, sobretudo, que esta linha seja correta (é fácil equivocar-se com seqüências de 12 cifras). Não esqueçam "FP" ao final. Utilizem números de linhas sucessivos aos do dicionário, deixando algo de separação (digamos 300) para possíveis (ou seja, seguros) acréscimos sucessivos.
4) Introduzam as linhas com os DATA das ações (a tabela

das ações controlando que cada uma compreenda número de síntese e número de ação. Verifiquem muito cuidadosamente que os números de síntese estejam em ordem correlativo, e não esqueçam "FA" ao final. Os números de linha devem ser sucessivos aos do plano, com a correspondente separação.

5) Introduzam as linhas com os DATA dos objetos, verificando que estejam na ordem desejada (que escreveram em algum lugar), que estejam em grupos de três (descrição, código de dicionário, lugar inicial) e que ao final não falte "FO". Vale a mesma explicação para os números de linhas.

6) Em continuação do intérprete (portanto antes dos DATA) escrevam as subrotinas de morto, de tempo e de introdução, deixando espaço para as modificações. Ainda que disponham em seu computador de RENUMBER, convém utilizá-lo o menos possível, porque lhes fará perder as referências às quais estão acostumados e para as que preparam as diferentes subrotinas. Usem o mesmo ao final para reordenar tudo. (Se estão entre os afortunados usuários do Microsoft Basic 2.0 no Macintosh, não terão problema dos nùmeros de linha). Situem as chamadas do parser nas linhas 720 e 780, na subrotina de introdução e na de tempo, respectivamente. Se não utilizam o tempo escrevam uma subrotina constituída por um simples RETURN (melhor não alterar o parser). Não se esqueçam, na introdução, de colocar em LU o número do lugar inicial.

7) Escrevam o comutador de ações sobre o modelo das linhas 1710-1750. A primeira linha (ON A GOTO...) indica os números de linhas das primeiras dez ações, a segunda (ON A-10 GOTO) os das dez sucessivas, e assim sucessivamente. A linha 1750 é muito útil durante a colocação em ponto do programa: se o número da ação a executar é maior que o número de linhas previstas no computador, imprime uma mensagem do tipo "AÇÃO 25" e volta. Para nossa comodidade, é melhor indicar os números de linha das ações com intervalos de 50, mais ou menos, de forma que fique um espaço suficiente entre um e outro.

8) Escrevam as subrotinas que executam as ações gerais, podem copiar das nossas (ações 1-7, linhas 1770-2120). Temos que modificar "Pega" (2) e "Deixa" (3), ficando os acréscimos que se referem à nave espacial. Também te-

mos que modificar "Save" (5) e "Load" (6) para seu computador, se não é um Apple II; não esqueçam de guardar e carregar também os possíveis temporizadores e/ou as variáveis especiais. Naturalmente, podem acrescentar as outras subrotinas gerais previstas por vocês (exemplo: "Move XXX", "Abre XXX", etc.): quantas mais tenha mais "inteligente" parecerá a aventura ao jogador. Importante:verifiquem,e se é necessário modifiquem, os números de linhas indicados no comutador. Cada um deles tem que corresponder à primeira linha da subrotina correspondente (e não a um REM). Em nosso programa, as ações 8 e 9 não foram usadas, com o fim de manter reagrupadas as ações gerais, deixando espaço para eventuais acréscimos. Organizem-se como quiserem.

9) Escrevam as subrotinas que executam as ações específicas de sua aventura, recordem sempre que tem que começar no número de linha indicado pelo comutador e na posição correspondente ao número da subrotina (o primeiro número indica onde começa a primeira subrotina, etc.).
10) Comprovem uma vez mais que todas as chamadas do comutador ON...GOTO vão à primeira linha da subrotina correspondente, e que cada subrotina acabe com RETURN ou com um GOTO a outra subrotina.

A aventura está terminada; não resta mais que comprová-la.

Eliminando erros (Debug)

Seguramente acreditaram que havíamos acabado, mas não estamos mais que na metade do trabalho (bom, um pouco mais da metade). Restam por fazer duas operações que requerem bastante tempo, mas que são fundamentais para o bom andamento de um jogo de aventuras: o debug e a avaliação.

O debug, como sabem, consiste na busca e correção de erros. No caso de uma aventura em BASIC, existem dois tipos: erros de programação e erros do jogo. Os primeiros são bem conhecidos: se manifestam com o mal afamado "SINTAX ERROR", com outras mensagens mais extravagantes ou criam efeitos estranhos, como os que pode causar a falta de um RETURN ao final de uma subrotina. A propósito, se depois de ter impresso "Um pouco de paciência..." o programa é bloqueado,assinalando um erro;pode

ser que o número indicado na mensagem de erro não seja efetivamente o da linha que o contém. De fato, um erro nos DATA pode ser assinalado, segundo os casos, na linha efetiva de DATA onde está o erro, em uma linha da DATA sucessiva, ou em uma linha da subrotina de leitura (1040-1070). Motivo demais para que se comprovem bem os DATA. Outro erro típico é a referência a um elemento inexistente de uma matriz. Se suas aventuras prevêem mais de 100 vocábulos no dicionário, 30 lugares, 150 ações ou 50 objetos, deverão modificar as linhas 970-1000 que estabelecem a dimensão máxima de cada matriz. Dado que isto não é um curso de BASIC, os outros possíveis erros do programa deverão ser encontrados por vocês mesmos.

Para encontrar os erros de jogo somente existe um método: jogar a aventura provando, dentro dos limites possíveis, todas as combinações de acontecimentos e situações. A primeira coisa que temos que fazer é o controle do plano: provem escolher desde cada lugar todas as direções, incluídas as não permitidas, comprovando que coincidam com as previstas em seu desenho.

Quando estiverem seguros de que o plano é correto, comprovem o dicionário: a aventura tem que reconhecer todas as palavras que foram incluídas. Isto serve também para comprovar que respeitaram a correta ordem alfabética nos DATA. Depois verifiquem que os objetos se encontram nos lugares onde devem estar, e comprovem que somente podem ser pegos os que devem ser pegos. Ao fazer isto também estarão comprovando a subrotina de "Pega". Chegamos ao ponto mais importante: a comprovação das ações. Peguem seus apontamentos e, partindo da primeira ação, provem todas as possíveis variantes de cada ação prevista. Não esqueçam nenhuma ("total, seguro que funciona"). Provem absolutamente tudo. Em particular, comprovem a fundo a "seqüência principal", ou seja, a cadeia de ações que leva à solução da aventura. Uma famosa casa de software norte-americana colocou em circulação uma aventura não resolúvel por causa de um erro deste estilo (e teve que substituir vários discos): acontece até nas melhores famílias.

Para acelerar o trabalho de comprovação servirá de ajuda o seguinte: podem parar o programa (com CTRL-C, STOP, BREAK ou como é chamado em seu computador), mudar o valor das variáveis (especialmente LU, por exemplo LU = 5) e fazer CONT. Desta maneira podem mudar o estado do jogo sem ter que fazer seqüências de ações complexas. Isto é particularmente cômodo no caso de aventuras de dimensões médias, onde é necessário muito tempo para conseguir um determinado objeto ou chegar a certo

lugar. Usem também o "Save" para guardar a situação antes de provar todas as variantes em um determinado lugar (e de passagem comprovem também "Save" e "Load"). Uma advertência: se o programa parar durante o INPUT (linha 800), em alguns computadores não é possível continuar com CONT. Deve-se escrever, ao contrário, GOTO 800 para continuar corretamente.

Quando estiverem bem seguros de que tudo funciona corretamente, estarão preparados para passar à segunda fase.

Avaliação

A avaliação é uma tarefa que vocês mesmos não podem fazer. Não estamos desvalorizando sua habilidade, somente queremos dizer que a avaliação de uma aventura (como qualquer jogo) não pode ser feita pelo autor. De fato. **a avaliação serve para estabelecer como é aceito o jogo por seus destinatários: os jogadores.** Muitos autores omitem este passo fundamental, e mandam às lojas jogos tecnicamente perfeitos porém enfadonhos ou não jogáveis (e não somente aventuras).

Para a avaliação **necessitarão de um ou mais amigos,** se possível que não sejam programadores (ou que não tenha visto seus trabalhos). Sente-os, um a um, diante do teclado, **façam começar o programa e observem como atuam. Regra fundamental: não digam nada!** Não devem ajudá-los a resolver o jogo; ao contrário, devem fazer apontamentos e escrever todas as frases que não haviam previsto mas que merecem uma resposta, as situações demasiadamente difíceis, as que são demasiadamente fáceis, os "Uf", que aborrecimento, etc. Descobrirão que esse problema não era tão fácil como parecia, que esse gracejo não era tão divertido, que a linha óbvia de comportamento somente era óbvia para você, e demais coisas do tipo. Descobrirão também que alguns resolvem o problema A em seguida e abandonam no problema B. Outros fazem exatamente o contrário. Se existe muitos que abandonam no problema C, é o caso de fazer algum ajuste.

Sobretudo, recordem que a finalidade de uma aventura é a diversão do jogador, e não a satisfação de dizer "Jamais ninguém a resolveu". Por outro lado, superar obstáculos muito fáceis não dá satisfação a quem joga. A única maneira para encontrar o equilíbrio justo é fazer jogar o máximo possível de pessoas, e calibrar cuidadosamente a dificuldade e as ajudas.

A título de exemplo, eis aqui as principais variantes que a avaliação demonstrou necessárias para "A nave espacial condenada":

- Aceitar também "Lê indicador", não somente "Olha indicador". O indicador não tem efeito sobre as ações, mas é fundamental para a compreensão da trama para o suspense: tem que ser visto (entre outras coisas, porisso tem dois).
- Acrescentar um objeto "Escadinha que baixa" no lugar 4 (final corredor S) que, obviamente, não pode ser pega. Muitos jogadores, de fato, não têm o costume de explorá-lo todo e demoraram em encontrar o reator (que têm que fazer compreender o perigo e dar pé ao quebra-cabeças).
- A ação final "Empurra alavanca" é muito dificíl para a média dos jogadores principiantes. Alguns, além disso, imaginam uma alavanca tipo freio de mão e dizem "Abaixo alavanca" e outras frases por estilo. Por outro lado, não temos que facilitar muito a solução para não tirar a satisfação de tê-la encontrado. Um compromisso possível pode consistir em prever a ação. "Olha alavanca" e a correspondente resposta: "É uma alavanca típica de quadro de comandos, que pode ser movida em duas direções". Se o jogador olha a alavanca, pior para ele. Nas instruções está escrito que "Olha" é uma operação fundamental. A propósito: se querem publicar uma aventura, escrevam três ou quatro páginas com instruções simples e claras, dirigidas a um leitor que não tenha visto em sua vida um jogo deste tipo.

Ao fazer acréscimos e modificações ao programa, recordem-se de respeitar a ordem alfabética no dicionário e o número nas sínteses das ações. Os lugares e os objetos novos, ao contrário, podem tranqüilamente ser acrescentados ao final dos já existentes (não no meio, porque senão mudam os números).

Finalmente

Agora estão em condições de escrever sua própria aventura. Se não têm algo claro, releiam com calma o capítulo correspondente, consultem as tabelas contidas no apêndice e estudem o programa de "A nave espacial condenada". O capítulo descreve o funcionamento do intérprete, e o posterior ilustra possíveis acréscimos e melhoras para suas aventuras, por exemplo gráficos e som.

CAPÍTULO XII

O INTÉRPRETE

Até agora dissemos: utilizem o intérprete sem preocupar-se como funciona. Mas esperamos que sejam o suficientemente curiosos para querer estudar detalhadamente seu mecanismo. É um programa em BASIC relativamente breve (umas 150 linhas, incluídos os comentários), mas ligeiramente sofisticado: algumas de suas carcterísticas provêm do programa "Aventura no castelo", outras provêm de nosso estudo da linguagem ADL-1 (ADventure Language 1). É, ou ao menos queria ser também, um exemplo de boa programação: subrotinas breves, cada uma das quais desempenha um encargo bem definido.

Se tem um computador com um BASIC decente descobrirão que muitas subrotinas podem ser simplificadas. Por outra parte, este progama foi escrito para ser portátil, ou seja, para funcionar também com os BASIC pobres (como o do Commodore 64). Em particular, a limitação de 80 caracteres por linha é bastante molesta.

Descreveremos o programa partindo desde o começo e seguindo o fluxo lógico, com regressões de vez em quando para descrever as diferentes subrotinas. Preparados? RUN!

Para começar

A linha 140 envia à 710, ou seja, ao programa principal ("main program" ou "main" simplesmente para os amigos). Por que o pro-

grama principal está no final? Por um motivo prático: na maior parte dos BASIC, o tempo necessário para encontrar a linha indicada em uma instrução GOTO ou GOSUB é tanto mais longo quanto mais adiante esteja a linha no programa (a busca é feita seqüencialmente, partindo desde o começo). Se deduz que para aumentar a velocidade de execução convém colocar as subrotinas mais utilizadas ao começo do programa. O programa principal, que não é chamado por ninguém, pode ficar tranqüilamente ao final ou quase. Na realidade, nossas subrotinas não estão dispostas na ordem mais conveniente, mas a clareza também tem sua importância.

O main (linha 70) chama em primeiro lugar a subrotina de inicialização ou preparação das variáveis, na linha 970. Esta começa por dimensionar as matrizes utilizadas para conter o dicionário, o plano, a tabela das ações e os dados dos objetos (970-1000). Se uma ou mais dimensões são insuficientes para a aventura que querem escrever, podem aumentá-las tranqüilamente (compativelmente com a memória disponível). Também seria possível ler nos DATA o número de elementos contidos em cada matriz (ex.: READ FD), mas preferimos não fazê-lo e especificar a dimensão com um número fixo (ex.: 100 para o dicionário). Isto é porque a maior parte dos compiladores BASIC não aceitam variáveis nos DIM, mas exigem um número (o que é um compilador BASIC? para que serve? Comentaremos no próximo capítulo).

As matrizes numéricas são todas de tipo inteiro (como indica o símbolo % que segue ao nome) para economizar espaço, exceto CA. De fato, cada elemento de uma matriz inteira ocupa habitualmente 2 bytes de memória, contra os 4-5 de uma variável numérica normal. CA não pode ser de tipo inteiro, porque tem que poder conter também números de seis cifras, e o conteúdo de uma variável inteira geralmente está limitado a 32767. Atenção: não foi dito que tem vantagens, em termos de velocidade, com o uso das variáveis inteiras. Por exemplo, no Apple II e o C-64 as operações com variáveis inteiras são, na realidade, mais lentas que as outras (o valor é convertido a formato normal, é executada a operação, e o resultado é reconvertido).

A linha 1010 estabelece o nome dos arquivos a utilizar para a gravação ("Save") da situação atual. A 1020 não tem nenhum efeito, simplesmente recorda que as quatro variáveis indicadas valem zero ao começo do programa (útil em caso de reler o mesmo depois de alguns meses). As linhas 1040-1070 lêem as informações das linhas de DATA e as colocam nas matrizes recém dimensionadas. Dado que são substancialmente idênticas, descrevemos uma: a 1040. Em primeiro lugar, se lê um dado na variável alfanumérica A$; se o dado é "FD", a leitura do dicionário está terminada e

o programa pode continuar. De outra maneira, ou seja o dado é distinto de "FD", a variável FD (número de dados em dicionário) é incrementada (existe um dado). O elemento número FD (número 1 neste caso) da matriz DZ$, ou seja, o primeiro vocábulo, é dado lido (A$), enquanto que o dado sucessivo, seu código, pode ser lido diretamente desde os DATA (com um READ) no correspondente elemento da matriz DZ. Tendo lido assim um vocábulo e seu código, o GOTO 1040 repete o ciclo desde o princípio (acaba somente quando encontra o final do dicionário, indicado pelo dado "FD").

O mesmo sistema é utilizado nas linhas 1050, 1060 e 1070, com uma particularidade na 1060: tendo lido um dado alfanumérico (cadeia) em, A$, e havendo esclarecido que não é "FA", temos que colocar na matriz CA não uma cadeia, mas um número. Dado que em A$ já foi lida uma cadeia, esta é transformada em número com a função VAL. A inicialização terminou: a 1080 volte ao programa principal, e também voltamos nós.

A 720 chama a uma subrotina (já vista) escrita pelo autor da aventura, que imprime os títulos, estabelece o lugar inicial (LU), prepara possíveis temporizadores e variáveis especiais, e volta. A 730 chama à cômoda subrotina de 1120 (Aperta < espaço > para continuar), que serve para evitar que as informações na tela "desapareçam" da vista antes de que o jogador tenha tido tempo de ler. A linha 1130, que espera a que se aperte o espaçador, depende da versão de BASIC, como já foi visto no segundo capítulo. Assim como está, serve para o Apple II e o Commodore.

O analisador sintático (parser)

As linhas 760-930 constituem o loop do jogo, ocupado em grande parte pelo parser (ou analisador sintático), que junto a suas subrotinas constitui a parte mais complexa do programa (aqui reside a "inteligência"). A Figura 1 mostra o diagrama de fluxos simplificado. Vamos seguí-lo em grandes passadas, antes de analisar seus detalhes mais interessantes.

Em primeiro lugar, é impressa a descrição do lugar no qual se encontra o aventureiro (760) e dos objetos presentes (770), depois é chamada a subrotina de passagem do tempo (780), já vista no capítulo 11. A linha 800 efetua a entrada da frase do jogador, e a 810 chama ao analisador do léxico, que envia o código da primeira palavra (verbo). Prévio controle (820) de que foi escrita efetivamente uma palavra (e não somente artigos ou espaços), a 830 é enfadada se a palavra acaba por "r" (provavelmente um verbo, como "pegar"), advertindo que devemos tutear ao com-

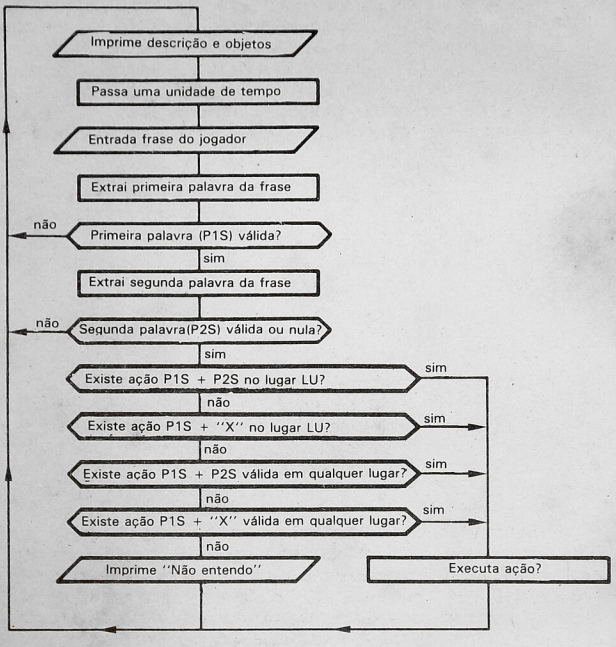

Figura 1. — Diagrama de fluxo simplificado do analisador sintático.

putador (por ex.: "pega"). A 840 verifica que a palavra está entre as do dicionário, e a 850 chama novamente ao analisador do léxico para conhecer o código da segunda palavra. Também esta é comprovada (860), recordando que pode ser nula (código zero) se a frase está formada por uma só palavra.

Se a segunda palavra existir, pode ser um objeto, a 870 procura o possível índice na pista dos objetos e o coloca em OB, para uso tanto do intérprete como do programador.

Já está tudo pronto: não resta mais que consultar a tabela

das ações para ver se a ação indicada foi prevista. A linha 880 prepara os dois primeiros terços da síntese de ação, para executar as multiplicações uma só vez ao invés de quatro. As linhas 890-920 procuram na tabela os quatro tipos possíveis de frase prevista, em ordem de precedência (leiam os REM). No caso de uma frase composta por uma só palavra, o código do nome (C2) é zero e são executadas somente a 890 e a 910 (as outras não teriam sentido).

O teste (IF A GOTO 760) ao final de cada linha volta ao princípio do loop no caso de que a subrotina 500 tenha reconhecido uma ação válida e a tenha executado em conseqüência (assinalando-a com A diferente a zero). Se não foi reconhecida nenhuma ação, a 930 imprime "Não entendo" e volta ao princípio do loop.

Análise do léxico

Vejamos alguns detalhes, começando pela subrotina da linha 280, chamada na 810 e na 850. Analisa a frase IN$ introduzida pelo jogador (com um comprimento de LI caracteres, ver linha 810) a partir do caracter número IN, extrai uma palavra e consulta o dicionário para estabelecer o código correspondente. Notem que o título da subrotina (260) resume o trabalho: a palavra é restituída em P$ e o código em C.

A linha 290 salta os possíveis espaços iniciais, até chegar ao princípio da palavra ou ao final da frase (neste caso MID$ restitui uma cadeia vazia). Se neste ponto o caracter a exame (o caracter número IN) se encontra mais além do final da frase não havia nenhuma palavra. Nesse caso, a 300 prevê a restituição de uma cadeia nula em P$. Tratando-se de uma subrotina não muito simples, preferimos saltar ao RETURN da 360 ao invés de colocar um RETURN na metade da subrotina, que sem dúvida teria causado problemas em caso de modificações sucessivas (disse o sábio: defendam-se das subrotinas com vários pontos de saída e, pior ainda, das que têm vários pontos de entrada).

A linha 310 assinala (em A) o ponto de início da palavra, linhas 320-330 a medem, caracter por caracter, até encontrar um separador, ou seja, um espaço, ou o final da frase. Neste ponto, "A" indica o primeiro caracter da palavra, e "IN" aponta ao separador que acaba a palavra. A 340 coloca a palavra em P$ e consulta o dicionário para determinar o código, a 350 descarta com desdém os artigos (código 7) e faz recomeçar a busca da palavra seguinte, como se o artigo não existisse.

A subrotina de busca no dicionário (180-240) não é mais que uma subrotina de busca binária: o dicionário é "aberto pela metade" e, após uma olhada, é descartada a metade que não pode conter a palavra (recordamos que estão em ordem alfabética); assim vai sendo partido pela metade, cada vez, a parte que pode contê-la. Em poucas passadas, ou a palavra é encontrada e a linha 190 restitui o código lendo-o na matriz correspondente DZ, ou não está no dicionário e a 230 restitui o código zero. "I" e "F" assinalam o começo e o final da zona do dicionário na qual ocorre a busca. A linha 180 os coloca nos limites do dicionário, depois a zona se estreita cada vez mais. Com 100 palavras, o vocábulo é encontrado (ou se sabe que não existe) em somente sete ciclos da subrotina.

Ações e objetos

Quando o analisador sintático já preparou uma síntese da frase e quer saber se existe (e eventualmente executar) a ação correspondente, chama a subrotina 500-550 (as chamadas estão nas linhas 890-920). Esta subrotina procura na tabela das ações, utilizando a subrotina 410-460 (que faz uma busca binária idêntica à do dicionário, recordam que as ações estão ordenadas por número de síntese?).

Se a ação não foi encontrada, a 500 volta com A = 0, e o parser passará a examinar a próxima possibilidade em ordem de precedência, ou imprimirá "Não entendo" se já as examinou todas. Se, ao contrário, a ação existe na tabela, A contém o número. Neste ponto pode ser que seja necessário comprovar a presença do objeto citado (OB): se o código de ação (A) é positivo, ou então, se não existe uma segunda palavra, a 510 salta esta comprovação. Em caso contrário (ou seja, se existe uma segunda palavra ou o código de ação é negativo — a 500 mostra que não pode ser zero—), a 520 comprova se o objeto está presente. Se OB vale zero não existe nenhum objeto e a subrotina volta com A = 1, isto é, como se tivesse sido executada corretamente uma ação. De fato, se foi impressa a frase "Aqui não está", o parser não tem que fazer posteriores buscas na tabela.

A ansiada linha 530, que executa a ação, é alcançada somente em dois casos:

- a ação foi encontrada na tabela e tem código positivo;
- a ação foi encontrada na tabela, tem código negativo e a segunda palavra se refere a um objeto presente.

No segundo caso, a linha 520 mudou de sinal o código, que, portanto, é positivo em cada caso. A 530 executa a ação "A" chamando ao comutador da linha 1710, e depois volta indicando "missão cumprida" (A = 1).

A subrotina 590-610 imprime a lista dos objetos presentes no lugar L, cada um precedido pela frase contida em P$. É utilizada pelo main (770) para enumerar os objetos existentes ao lado do aventureiro, precedidos de "Vejo" (exemplo: "Vejo um pressionador vermelho"), e pela subrotina de inventário (2120) para enumerar os objetos presentes no lugar zero, ou seja, possuídos pelo aventureiro. A subrotina é limitada a comprovar a lista dos objetos para verificar se o lugar onde se encontra cada um deles coincide com L. A função ABS serve para ignorar o possível sinal menos diante do número do lugar, que significa "que não pode ser pego" (por exemplo: a cama).

A breve mas importante subrotina 650-670, chamada pelo parser na 870, é a que coloca em OB o número (índice na matriz) do objeto citado como segunda palavra, mas somente se este está presente ou transportado (se não, coloca OB = 0). Funciona de maneira similar à anterior, investigando a lista dos objetos para ver se encontra um que corresponda às condições requeridas: código em dicionário igual a C2 e lugar igual a LU ou a zero. Sem esta subrotina seria impossível referir-se a um objeto na forma IF LO% (OB)...

CAPÍTULO XIII

VARIAÇÕES SOBRE UM MESMO TEMA

Compilar

Este último capítulo está dedicado às possíveis variações, acréscimos e melhoras que podem ser feitas em um programa de aventuras. A primeira e mais simples operação consiste em compilar o programa BASIC, isto é, transformá-lo em um programa equivalente, em código máquina ou em outra forma mais eficiente. Este trabalho o fazem automaticamente certos programas chamados, precisamente, compiladores BASIC.

São programas complicados que, certamente, requerem a utilização de uma unidade de disco (disk drive). Para os computadores de maior difusão existe, pelo menos, um compilador BASIC, e para alguns existe uma ampla gama.

As vantagens obtidas de um compilador são essencialmente duas:

- a velocidade de execução aumenta notavelmente (inclusive mais de 10 vezes);
- o programa original (ou "fonte") já não é legível

A primeira vantagem começa a ser notada com aventuras de um certo tamanho e, sobretudo, com muitos objetos. A segunda é talvez mais importante: o programa não pode ser lido e modificado por qualquer um, e o trabalho fica protegido (não da cópia, obviamente). Além disso, todos os comentários são eliminados pelo programa e não ocupam espaço no resultado final. Como com-

pensação, quase todos os compiladores aumentam consideravelmente as dimensões do programa: se têm pouca memória disponível, seguramente terão problemas.

Considerando que os compiladores diferem profundamente um do outro, proporcionamos uma regra empírica: para um programa de comprimento médio, as dimensões do código compilado são mais ou menos iguais às do programa original comentado amplamente. Partindo de um programa sem REM (ou seja, muito mal escrito) as dimensões aumentam, geralmente, desde um mínimo de 20-30% até o dobro do original. Se não quiser ou não puder compilar o programa, pelo menos pode usar uma utilidade de "crunch" (compactação, compressão). Se trata de um programa que, como mínimo, tira todos os REM e une o máximo de linhas possíveis em uma só linha BASIC. Assim, o programa é feito muito menor, ligeiramente mais rápido na execução e, sobretudo, de difícil leitura.

Em ambos os casos, compilação ou compactação, comentem amplamente a versão inicial do programa. Dado que os comentários serão tirados não lhes custa quase nada e simplifica muito o trabalho, especialmente se tem modificações.

Melhoras do intérprete

O programa pode ser melhorado de muitas maneiras. Por exemplo, para permitir guardar (em disco ou em fita) várias situações com nomes diferentes. Basta modificar adequadamente as subrotinas de "Save" e de "Load". Além disso, seria a ocasião para ter em conta os possíveis erros durante a gravação e posterior leitura (por exemplo: disco completo, disco protegido, disco deteriorado, etc.), evitando incômodas paradas do programa. Uma advertência para quem utiliza o Apple II com Prodos: não utilizem a ordem STORE para guardar as variáveis. As matrizes de cadeia não lidas pelos DATA introduzem periódicas e antipáticas pausas no programa. Deixem as coisas como estão.

Sentindo-se programadores experts, provem a acrescentar um controle do número máximo de objetos transportáveis. Parece fácil, mas não basta contar os objetos pegos com "Pega" e os deixados com "Deixa". Temos que ter em conta todas as mudanças de lugar, desaparições, etc. Naturalmente, não queiram fazer este controle cada vez que um objeto seja deslocado em uma subrotina de ação (exemplo: LO%(7) = 13). Tratem de inventar algo melhor.

Alguns jogadores preferem escrever em primeira pessoa (exemplo: "Aperto o botão". Nós achamos mais simpática a segunda pessoa, porque também introduz um tanto de esquizofrenia (dissociação mental) e conseqüentes conflitos psíquicos entre "a mente" (o jogador) e "o braço" (o personagem) que contribuem à diversão. Se de todas as formas quiserem a primeira pessoa, apenas devem mudar os verbos no dicionário. Cuidado com os reflexivos ("Me atiro"): tem que ignorar o "Me". Também podem ignorar a última letra de um verbo, porém tenham cuidado com os irregulares (ex.: "Fazes", "Faço").

Finalmente, uma possibilidade interessante: o fato de algumas ações pode depender também do azar (graças à função RND). Como sempre, se trata de uma arma de duplo fio: se a coisa não está bem estudada, são obtidos comportamentos completamente idiotas. Se é usado com parcimônia, pode contribuir à animação do jogo.

Cor e som

Se seu computador pode escrever a cores na tela, têm a ocasião de acrescentar um pouco de variedade e animação às mensagens impressas pelo programa. Podem utilizar uma cor para as descrições, outra para os objetos, uma distinta para as entradas e outra para as mensagens, reservando imprevistas mudanças de cor ou de fundo para situações especiais (por ex., fundo preto no espaço). É fácil modificar o programa para esse fim. Basta inserir as instruções para a mudança da cor dos caracteres (que geralmente são diferentes entre um computador e outro) nos pontos apropriados:

- para as descrições, ao começo da linha 760;
- para os objetos, ao começo da linha 590;
- para as entradas, ao começo da linha 800.

Viram a vantagem de desenvolver um determinado trabalho em um só ponto do programa? As modificações são muito mais simples. Para as mensagens, podem mudar a cor depois de 800 ou colocá-la sempre em seu lugar depois de ter modificado nos três casos anteriores. Também podem distinguir as mensagens do parser (tipo "Não entendo", "Não o tem", etc.) dos do jogo, mudando a cor ao começo do comutador de ações (1710), mas entendemos que o resultado não é dos melhores, desde o ponto de vista do jogo.

À parte da cor, podem introduzir algum efeito sonoro nos pontos mais significativos da aventura (por exemplo, um assobio do ar no compartimento estanque ou o apito de alarme do reator). Não encham a aventura de sons e ruídos: fazem muito mais efeito se chegam inesperadamente.

Desenhos e gráficos

Desde o ponto de vista do programa, uma aventura gráfica é idêntica a uma de tipo texto, como a que está descrita neste livro. A única diferença é que as descrições dos lugares foram substituídas por desenhos, assim como também (nas aventuras bem feitas) as descrições dos objetos. Em compensação, as mensagens de resposta geralmente são muito breves, tendo que estar na parte da tela não ocupada pelas figuras.

Se existe em circulação muitas aventuras de tipo texto mal escritas, de tipo gráfico mal realizadas, existe uma verdadeira invasão, especialmente para os computadores domésticos mais difundidos. Muitos acreditam que algum desenho (freqüentemente aproximado) basta para tapar uma substancial carência de inteligência do programa e de fantasia do autor. São particularmente incômodas as aventuras cujos problemas são resolvidos com uma só frase, escrita com as duas únicas palavras admitidas (não funciona nenhum sinônimo). Já falamos disto, mas temos a impressão de que muitos autores utilizam os gráficos como simples tampão de trabalhos de baixa qualidade. Moral: Se não acreditarem que são capazes de escrever uma boa aventura gráfica, renunciem aos desenhos e procurem a maneira para que os jogadores se divirtam. Se, ao contrário, têm ânimo, damos algumas idéias sobre as quais trabalhar:

- em primeiro lugar, considerem que em quase todos os computadores a utilização dos gráficos tira memória do BASIC. Tenham isto em conta adequadamente;
- uma maneira pouco comprometida, mas passivelmente eficaz, para inserir gráficos em suas aventuras consiste em fazer aparecer, de vez em quando, imagens "em toda tela" das cenas mais significativas (por exemplo, a sala do reator da nave espacial). Simplesmente se trata de tomar a imagem do disco quando for necessária; se não tem disk drive, ou este é tão lento como o homem, esqueçam-se dele (se são experts, podem trabalhar em linguagem máquina e ter al-

guma imagem nos "buracos escondidos" da memória). A quantidade de imagens pode ser aumentada se tem à sua disposição um programa de utilidade que lhes permita ter as imagens em formato comprimido (isto é, utilizando menos memória).

- um método mais sofisticado, utilizado em grande parte das melhores aventuras, consiste em utilizar uma "Linguagem gráfica", na qual a figura está descrita como um conjunto de pontos, linhas, áreas, e assim sucessivamente. Por exemplo, um retângulo é memorizado simplesmente com as coordenadas de dois vértices opostos. Este tipo de linguagem, que permite ter muitas imagens em pouca memória, já pode estar disponível no computador em forma mais (Macintosh) ou menos (IBM, MSX) evoluída, pode ser adquirido em disco (exemplo: Graphics Magician para Apple II) ou pode estar totalmente escrito por vocês mesmos se tiverem a paciência e o conhecimento necessários para fazê-lo. Se requer, como mínimo, que o computador tenha as instruções para desenhar pontos e traçar linhas.
- nas aventuras gráficas o problema principal está constituído pelos objetos: fica antipático desenhar um lugar sem mostrar os objetos presentes, ou afrontar um árduo caminho de superpor objetos sobre o desenho com perspectivas absurdas, ou limitar-se a um desenho dos objetos presentes fora da figura principal. Para o inventário, a figura pode ser substituída por figuras menores representativas dos objetos transportados.

Aqui paramos. Apenas arranhamos a superfície do problema; necessitaríamos outro livro para tratá-lo de maneira adequada (poderíamos também escrevê-lo; antes ou depois...). Em essência, o caminho das aventuras gráficas pode dar grandes satisfações e grandes desilusões. Se o querem percorrer, primeiro pratiquem com aventuras clássicas de tipo texto. Recordem que os gráficos não podem nunca salvar uma aventura medíocre.

Conclusão

Aqui termina a aventura. Tentamos passar uma parte de nossa experiência, agora é a sua vez. Por nossa parte, temos muitas coisas que contar-lhes e esperamos ter logo a ocasião de fazê-lo.

APÊNDICE A

VARIÁVEIS PRINCIPAIS DO INTÉRPRETE

Dicionário:

FD	Número de vocábulos no dicionário.
DZ$(1..FD)	Vocábulos em ordem alfabética.
DZ(1..FD)	Código dos vocábulos correspondentes a DZ$.

Plano:

FP	Número de lugares no plano.
DE$(1..FP)	Descrição dos lugares em ordem de número.
DI$(1..FP)	Conexões do lugar descrito no correspondente elemento de DE$ com os seis lugares adjacentes (N/S/E/O/A/B).

Ações:

FA	Número de ações na tabela.
CA(1..FA)	Códigos (síntese) de ação em ordem numérica.
AZ%(1..FA)	Número das ações correspondentes a executar; negativo se é requerida a presença do objeto citado.

Objetos:

FO	Número de objetos.
OB$(1..FO)	Descrição dos objetos em ordem do número de objetos.

CO%(1..FO)	Código no dicionário do objeto descrito no correspondente elemento de OB$.
LO%(1..FO)	Lugar do objeto correspondente:
	1..FP lugar indicado, pode ser pego.
	-1..-FP lugar indicado, não pode ser pego.
	0 transportado pelo aventureiro.
	-99 "limbo" (fora do jogo).

Várias:

LU	Lugar atual do aventureiro (variável).
OB	Número do objeto citado como segunda palavra; zero se não está presente ou é transportado.
P1$,P2$	Primeira e segunda palavras válidas da frase.
C1,C2	Códigos no dicionário de P1$ e P2$.

APÊNDICE B

SÍNTESE DE AÇÃO E PRIORIDADES DE BUSCA

Síntese de ação = número de 6 cifras (LLPPSS):
Código do lugar (2) + Código primeira palavra (2) + Código segunda palavra (2).
Exemplo 034709: a ação é executada se a frase composta por duas palavras com códigos de dicionário 47 e 9 é escrita no lugar 3.

- **Primeiras duas cifras (código lugar):**
  00 Ação válida em qualquer lugar.
  01..FP Ação válida somente no lugar indicado.

- **Cifras centrais (código primeira palavra):**
  01..FD Código da primeira palavra no dicionário.

- **Últimas duas cifras (código segunda palavra):**
  00 Segunda palavra nula, frase de uma só palavra.
  01..FD Código da segunda palavra no dicionário.
  99 (X). Vale qualquer palavra sempre que seja válida, ou seja, existente no dicionário e não ignorada (ex.: artigos).

Prioridades de busca de ações com frases de uma palavra:
LLPPOO Palavra escrita no lugar LL.
OOPPOO Palavra escrita em um lugar qualquer.

Prioridades de busca de ações com frases de duas palavras:

LLPPSS Par de palavras pronunciadas no lugar LL.

LLPP99 Primeira palavra, seguida de uma palavra válida qualquer, pronunciada no lugar LL.

OOPPSS Par de palavras pronunciadas em um lugar qualquer.

OOPP99 Primeira palavra, seguida de qualquer palavra válida, pronunciada em qualquer lugar.

NOTAS

A realização e utilização dos chamados "jogos de aventuras" é, sem dúvida, um dos aspectos mais atraentes dos computadores. Com este livro da B.B.I. oferecemos simultaneamente a possibilidade de acesso a:

- um jogo de aventuras completo;
- a técnica para desenhar suas próprias aventuras sem excessivo trabalho;
- novos conhecimentos sobre o BASIC.

Para conseguí-los nos baseamos no grande jogo "A nave espacial condenada". Após apresentá-lo e oferecer sua listagem o leitor é conduzido mais além do cenário e das lamparinas, passando sobre os PRINT e os READ, para explicar e ensinar os truques mais úteis e curiosos que são empregados neste gênero.

É suficiente um mínimo conhecimento do BASIC para ser capaz de desenhar a partir daqui seus próprios jogos de aventuras, aproveitando o "esqueleto" universal que incluimos. Os mais experts encontrarão também uma descrição completa do funcionamento do programa, o que lhes dará a oportunidade de construir jogos mais complexos e sofisticados.

Cz$ 49.50

17

DESENHO DE JOGOS

B.B.I.