O FORTH possui uma estrutura bastante diferenciada das outras linguagens. Costuma ser denominado "linguagem inacabada", visto proporcionar uma liberdade quase total de criação de novas palavras (comandos) e sua incorporação à estrutura da linguagem. Esta flexibilidade, aliada à facilidade da técnica de programação TOP-DOWN que sua estrutura permite, tem possibilitado variadas aplicações.

O sistema GraFORTH, além dos recursos da linguagem FORTH, incorpora muitas facilidades dos comandos TURTLE GRAPHICS da linguagem LOGO, mantendo ainda no sistema numerosos editores de gráficos e animação, ferramentas indispensáveis para a confecção de um programa mais bem elaborado. O GraFORTH é, assim, muito mais do que um elemento nas longas listagens de linguagens, de editores de gráficos bi e tridimensionais, de processadores de textos ou de programas de animação gráfica disponíveis para os computadores pessoais da série Apple II (II, II Plus, IIe, enhanced IIe, IIc e compatíveis).

A presente obra pretende apresentar tanto aos usuários experientes como aos iniciantes as melhores formas de um real aproveitamento dos recursos do GraFORTH, provendo uma vasta gama de informações úteis e curiosas.

hemus

O SISTEMA GraFORTH — JAMES SHEN — GILBERTO M. MARTINS

hemus

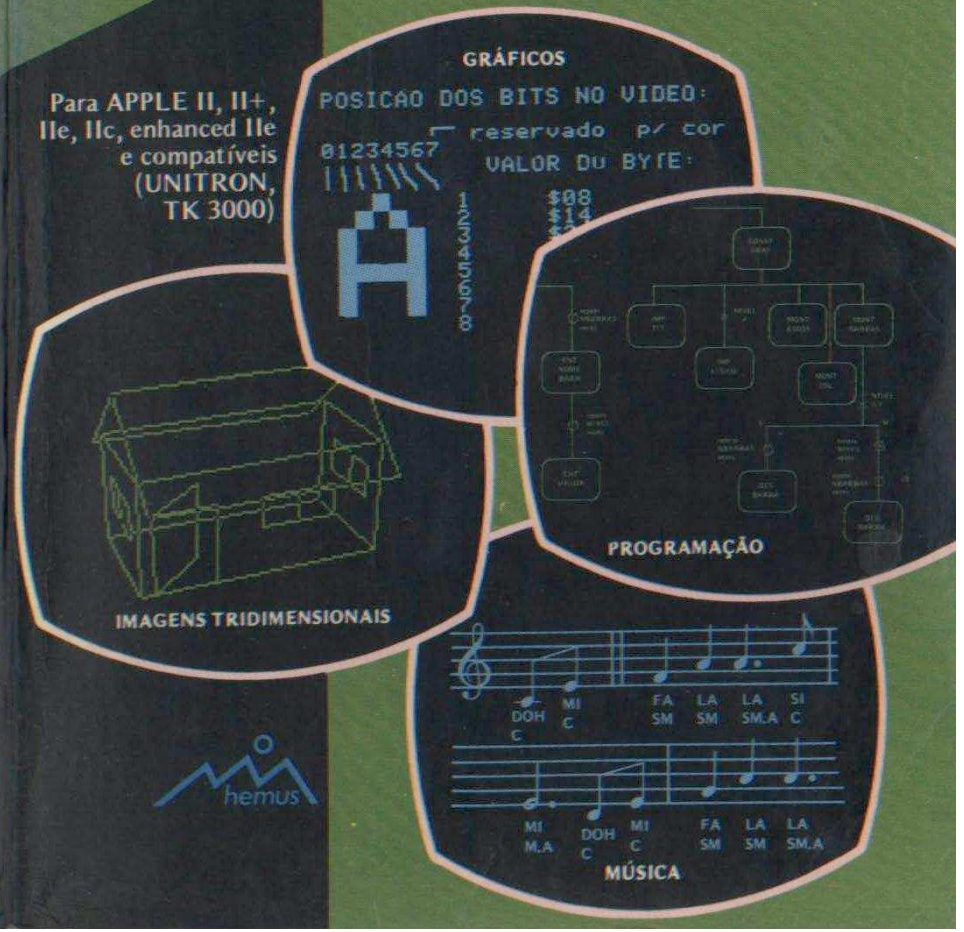
Série Programação Avançada em APPLE

JAMES SHEN
GILBERTO MOREIRA MARTINS

O SISTEMA GraFORTH

Programação e Animação Gráfica

Para APPLE II, II+, IIe, IIc, enhanced IIe e compatíveis (UNITRON, TK 3000)



hemus

O SISTEMA

GraFORTH

PROGRAMAÇÃO
E ANIMAÇÃO GRÁFICA

James Shen
e Gilberto Moreira Martins

O SISTEMA GraFORTH

PROGRAMAÇÃO E ANIMAÇÃO GRÁFICA

hemus

Revisão:
Orlando Parolini
e
Edilson Navas

Composição e arte:
Estúdio Behar

Capa:
Carlos das Neves

hemus

hemus editora limitada

01510 rua da gloria 312 liberdade caixa postal 9686
fone 2799911 pabx telex (011) 32005 edit br
endereco telegrafico hetec são paulo sp brasil

Impresso no Brasil/*Printed in Brazil*

Nota do Autor

Sem dúvida, a ciência da computação e a tecnologia da informação proporcionaram a mais ampla e séria repercussão em quase todas as atividades manuais e intelectuais do homem nestas últimas décadas. Queiramos ou não, havemos de conviver com a proliferação contínua da Informática.

Pensando na importância de o homem poder dominar a Informática em benefício da própria Humanidade, continuamente tentaremos divulgar as mais variadas informações possíveis sobre a ciência da computação e a tecnologia da informação. Seguindo esta nossa filosofia editorial, publicamos o presente livro.

Esta obra é o primeiro livro da série Programação Avançada em Apple, que visa fornecer aos usuários de um sistema Apple II (II, II Plus, IIe, enhanced IIe, IIc ou compatíveis) as melhores alternativas para a utilização de seu sistema microcomputador em função das suas próprias aplicações e necessidades.

Desejamos agradecer a todos os leitores que acompanharam todas as nossas publicações ou algumas delas na área de Informática até o presente momento. Esperamos que continuem nos prestigiando, e aguardando que a Hemus Editora coloque em breve no mercado as novidades e as surpresas que a todos irão deslumbrar.

James Shen

Consultor em Marketing e Sistemas
Coordenador das séries:
Guia do Programador
Programação Avançada em Apple

Agradecimentos

Os autores desejam agradecer os preciosos comentários, sugestões e incentivos de seus amigos e dos funcionários da Hemus Editora Ltda., sem os quais a idéia de escrever este livro não se concretizaria.

Agradecemos especialmente a MAXIM BEHAR e a URI BEHAR, diretores desta editora, por sua assistência sempre atenciosa a nossos trabalhos, seu apreço e respeito à nossa capacidade profissional, por seu apoio quase contínuo às nossas pesquisas e, principalmente, por sua valiosa amizade.

Dedicamos este livro aos nossos familiares e amigos que sempre nos incentivaram a adquirir cada vez mais o gosto pelo estudo, pela pesquisa e pelo trabalho.

Os autores

Índice

Prefácio

É inegável que hoje a sociedade está aceleradamente se informatizando cada vez mais. Tal evolução nos atinge pelas mais variadas formas possíveis na nossa vida cotidiana. Chegamos então à conclusão de que, queiramos ou não, vamos incorporando as novas maneiras de ver, sentir e ser, ditadas pela proliferação da Informática. E um indício disso pode ser a nossa crescente preocupação quase automática em adquirir mais conhecimentos sobre esse "admirável" mundo da Informática.

Se você adquiriu (ou pretende adquirir) um microcomputador da família Apple II (II, II Plus, IIe, enhanced IIe, IIc ou compatíveis), perceba que essa sua decisão também é um modo, consciente ou não, de você demonstrar tal preocupação. Um sistema Apple II é muito poderoso e versátil em termos de hardware (máquina) e software (programas), e se a sua intenção inicial era a de melhor utilizar a potencialidade de um microcomputador em sua aprendizagem e em sua "adaptação" a um mundo novo por vir, temos plena certeza de que você já se encontra no caminho certo, visto que o Apple II é um dos melhores e mais poderosos sistemas de microcomputadores existentes no mercado. Este livro é, assim, uma forma de dar-lhe nossas boas-vindas ao fascinante mundo do Apple II.

Esta obra tenciona servir como um guia de orientação tanto aos programadores experientes quanto aos usuários iniciantes na Microinformática. Ela vem satisfazer a muitos dos interesses dos leitores no sentido de poder provê-los de uma gama bem variada de informações úteis e curiosas para sua formação cultural, a fim de que possam implementar sistemas de aplicações adequados ao hardware do seu sistema Apple II. Contudo, é preciso ressaltar que determinados conhecimentos prévios são exigidos para se aproveitar desse livro ao máximo. Supomos que o leitor deve ter um capital cultural razoável, pois, no decorrer do livro, aqueles menos informados poderão sentir uma falta preciosa de vários conceitos matemáticos. Esforçamo-nos demais para fazer com que todas as informações transmitidas fossem muito mais do que um monte de dados confusos. Para tanto, colocamos muitos exemplos práticos; tentamos provar, sempre que possível, os conceitos teóricos com explicações e críticas comparativas;

optamos por uma filosofia de ensino que desse ao leitor uma aprendizagem passo a passo; e planejamos cuidadosamente toda uma estrutura de contínua interação com os leitores para explicar todas as suas possíveis dúvidas.

Acreditamos que a obra aqui apresentada será bem acolhida pelo público leitor em geral, uma vez que os assuntos tratados despertam em muitos um encanto fascinante. Quem não se deslumbra com o poder de programar numa linguagem versátil, e com o poder de criar e de animar gráficos bi ou tridimensionais no seu sistema Apple II? Diria um amigo nosso: "tudo isso é muito estupendo!".

Confessamos ter sentido um enorme entusiasmo pelo sistema GraFORTH. Nossa admiração era tal, que chegamos a compartilhar nosso entusiasmo com vários amigos, os quais muito nos incentivaram a escrever o presente trabalho.

Constituído de nossos esforços coordenados de pesquisa e de contínuos incentivos, o resultado aqui está, depois de oito meses de árduo trabalho de planejamento e desenvolvimento.

Aguardamos que os leitores venham nos fornecer suas preciosas contribuições a fim de que possamos aperfeiçoar cada vez mais o nosso trabalho, enviando suas opiniões, críticas e sugestões para a CAIXA POSTAL 9686 – São Paulo – SP – CEP 01051.

JAMES SHEN
GILBERTO MOREIRA MARTINS

Introdução

O que você deve saber

Se você deseja aprender a programar numa linguagem tal como o FORTH, especificamente o GraFORTH, deve ter em mente que isso requer um pouco de paciência e prática, mas no final da sua animada aprendizagem, todo seu esforço será recompensado.

O sistema GraFORTH é muito mais do que um elemento nas longas listagens de linguagens, de editores de gráficos, de processadores de textos ou de programas de animação disponíveis para o Apple II. O GraFORTH é, antes de tudo, o resultado de um esforço persistente em tornar seu sistema Apple II muito mais produtivo, eficiente e habilidoso.

O microcomputador da família Apple II (II, II Plus, IIe, enhanced IIe, IIc ou compatíveis) apresenta poderosas capacidades gráficas que são muito mal utilizadas pelas linguagens disponíveis, principalmente pelos interpretadores Applesoft BASIC e Integer BASIC. Julga-se que uma forma de contornar esse problema é usar, diretamente, os recursos da máquina com a linguagem Assembly. Programar, contudo, em Assembly é mais um problema do que uma solução propriamente dita. O sistema GraFORTH vem se tornando, assim, e cada dia mais, uma ferramenta versátil e indispensável para o tratamento de gráficos de alta resolução no seu sistema Apple II.

O GraFORTH é uma linguagem desenvolvida a partir de uma filosofia de comparar um objeto desconhecido com um outro identificável por nós, facilitando em muito a nossa assimilação pela linguagem. Tentamos, assim, fazer jus aos recursos do sistema GraFORTH.

A estrutura do FORTH associa-se à programação estruturada, pois programar em FORTH é utilizar as técnicas da programação estruturada. E o FORTH tem na sua linguagem os conceitos da notação polonesa inversa, da pilha de dicionário de palavras e da pilha de retorno dos dados, os quais tornam esta linguagem flexível, versátil e capaz de adaptar-se às mais diversas aplicações particulares. O GraFORTH, sem dúvida, aproveita muito bem todos esses recursos, além de juntar, ao FORTH padrão, sua característica mais importante: a velocidade.

O que você deve ter

Para um melhor acompanhamento e aproveitamento deste curso de programação de linguagens e de animação gráfica, é necessário que o leitor tenha acesso aos seguintes componentes de hardware e software: um microcomputador da família Apple II (II, II+, IIe, IIc, enhanced IIe ou compatíveis) com um mínimo de 48 Kbytes de memória (porém, o ideal é de 64Kb), uma unidade de disco (disk drive) operando com sistema operacional DOS 3.3 ou compatível, um monitor de vídeo (ou uma televisão com modulador RF) monocromático ou colorido, e um disquete contendo o sistema GraFORTH.

O disquete original do sistema GraFORTH possui normalmente os seguintes arquivos catalogados no diretório e mostrados pelo comando CATALOG do sistema operacional DOS 3.3 do Apple II:

```
DISK VOLUME 254

B 037 OBJ.FORTH
B 011 OBJ.EDITOR1
B 011 OBJ.EDITOR2
T 021 CHAREDITOR
T 024 IMAGEDITOR
T 015 PROFILE
T 004 TURTLE
T 022 PLAY
T 004 STRING WORDS
A 002 A ********** CHARACTER SETS **
B 005 CHR.SYS
B 005 CHR.STOP
B 005 CHR.SLANT
B 005 CHR.GOTHIC
B 005 CHR.BYTE
B 005 CHR.STUFF
B 005 CHR.MAXWELL
A 002 A ***** GRAFORTH DEMO FILES **
T 002 QUERY
T 013 HEADER
T 008 MENU
T 008 GRAPHICS1
T 010 GRAPHICS2
T 016 GRAPHICS3
T 012 TEXTDEMO
T 018 FORTHDESC
T 011 FLEDERMAUS
T 011 PIANO
T 004 CLOCK
```



```
A 002 A ********** 3-D IMAGES **
B 002 TETRA
B 002 XYZ
B 002 BAT
B 002 CUBE
B 002 HOUSE
B 010 CHAL
T 003 BIGCHAL
```

Como usar este livro

Este livro foi elaborado visando explicar melhor as aplicações possíveis do sistema GraFORTH, muito embora o manual do Insoft, Inc. sobre o GraFORTH seja bem escrito, porém, pouco acessível aos usuários iniciantes na Informática e insuficiente aos programadores mais experientes. Este livro então será uma estrada que você terá de percorrer antes de chegar ao seu destino. Nosso objetivo é auxiliá-lo a compreender melhor todo o sistema GraFORTH. Contudo, você poderá consultar qualquer um dos itens na ordem que quiser.

Parte primeira

Apresentando o sistema GraFORTH

1

INTRODUÇÃO AO SISTEMA GRAFORTH

1.0 O sistema Apple II

O seu Apple II é um poderoso microcomputador que possibilita as mais variadas aplicações profissionais, educacionais e domésticas. Uma das principais características do sistema Apple II, que explicaria todo o seu sucesso comercial e toda a sua popularidade, é a sua modularidade, a significar que o usuário pode configurar o seu sistema Apple II de acordo com as suas necessidades. Como um exemplo de configuração, observe que o sistema Apple II mínimo requerido pelo GraFORTH consiste do próprio Apple II (II Plus, IIe, enhanced IIe, IIc ou compatíveis), com pelo menos 48 Kbytes (sendo que o ideal é de 64Kb), monitor de vídeo (ou televisão com modulador RF) monocromático ou colorido, de uma unidade de disco (disk drive). Uma outra característica atraente do Apple II é sua capacidade gráfica que é ampla e racionalmente explorada pelo sistema GraFORTH. Sem dúvida, os microcomputadores da família Apple II foram marcos importantes na história dos computadores.

1.1 A linguagem FORTH

Charles H. Moore desejou criar uma linguagem flexível que viesse contribuir para melhorar a interação homem-máquina. A implementação dessa linguagem ocorreu num computador IBM 1130 no início dos anos 70. Entusiasmado com os recursos dessa linguagem, Moore a denominou de FOURTH (quarta geração). Por capricho do destino (o IBM 1130 aceitava somente cinco caracteres no seu identificador), Moore teve então de alterar o nome FOURTH, e essa linguagem ficou, assim, conhecida como FORTH. Desde então, surgiram as variadas versões de FORTH, das quais o GraFORTH é uma versão especialmente desenvolvida para os programadores e/ou usuários de um sistema Apple II.

O FORTH possui uma estrutura bastante diferenciada das outras linguagens. Foi, muitas vezes, denominado de "linguagem inacabada", visto que você tem quase total liberdade de criar novas palavras (comandos) e incorporá-las na estrutura da linguagem. E essa flexibilidade, aliada à facilidade da técnica de programação permitida por sua estrutura, possibilitou que a linguagem FORTH fosse usada nas mais diferentes aplicações.

1.2 O sistema GraFORTH

O sistema GraFORTH é um conjunto de programas que providencia uma série de recursos só agora disponíveis para seu sistema Apple II. Agora você pode desenhar e animar gráficos tridimensionais (coloridos ou não) numa velocidade de execução só comparável aos programas escritos em Assembly, prover nos seus programas sons e música, exibir "textos" com caracteres personalizados ou nos estilos mais variados (itálicos, góticos etc.), gerar novas palavras interiores à linguagem de acordo com suas necessidades etc.

Em relação às outras versões de FORTH, o sistema GraFORTH possui muitas características inovadoras e diferentes, embora a estrutura geral da linguagem fosse mantida. Tais modificações foram necessárias para que o usuário pudesse melhor aproveitar os recursos do Apple II em qualquer situação possível. Ressaltamos, novamente, que o sistema GraFORTH foi especificamente desenvolvido para os mais variados sistemas Apple II.

Além dos recursos do FORTH, o sistema GraFORTH incorpora muitas das facilidades dos comandos TURTLEGRAPHICS da linguagem LOGO, mantendo ainda no sistema os mais variados editores de gráficos e animação, os quais são ferramentas indispensáveis na confecção de um programa melhor elaborado.

O sistema GraFORTH é, assim, um sistema integrado de compilador, editor de textos, editor de gráficos tridimensionais, editor de caracteres e programas demonstrativos.

1.3 Como editar seus programas

O editor de textos (processador de textos) do sistema GraFORTH permite-lhe entrar e editar seus programas-fontes, armazenando-os num arquivo de textos acessível pelo sistema operacional DOS 3.3 para posterior compilação. Você pode usar também alguns dos processadores de textos disponíveis para o sistema Apple II, desde que eles guardem num arquivo de textos o programa assim entrado e editado.

O sistema GraFORTH também possibilita que você entre diretamente uma palavra ou blocos de palavras, que são compilados assim que são entrados pelo sistema, podendo ser imediatamente executados ou não. Como o sistema GraFORTH compila imediatamente as palavras ou blocos de palavras entrados, armazenando na memória só programa-objeto, sem se preocupar com o programa-fonte, é deveras importante que você venha a entrar e editar seu programa num editor de textos e salvar o programa-fonte num disquete para posterior correção, caso haja erros de compilação (erros de sintaxe) ou erros de lógica (erros de execução). (Ver o Capítulo 11 para obter maiores informações de como usar o editor de texto do sistema GraFORTH.)

1.4 Comparando o GraFORTH com o TransFORTH

Muitas das características do GraFORTH são conservadas no sistema TransFORTH. Esta é uma versão diferenciada do GraFORTH, com a filosofia de orientar especificamente os usuários nos cálculos científicos e comerciais, possibilitando aritmética com números em ponto flutuante (números reais) e possuindo somente capacidade de criar gráficos bidimensionais, enquanto que o sistema GraFORTH possui a capacidade de criação e animação de gráficos tridimensionais e a restrição de calcular com números inteiros. Essa restrição foi propositalmente arquitetada no GraFORTH a fim de aumentar sua velocidade de execução, o máximo possível, para que pudesse desenhar e animar seus gráficos.

1.5 Carregando o sistema GraFORTH

Você pode inicializar o sistema GraFORTH a partir do Applesoft BASIC ou do Integer BASIC, entrando o seguinte comando: IN#6 ou PR#6, o qual faz com que o disquete com o sistema GraFORTH (juntamente com o seu sistema operacional compatível com o DOS 3.3) seja carregado para a memória do seu sistema Apple II.

Se o sistema operacional DOS 3.3 estiver ativo na memória, você pode entrar o comando: BRUN OBJ.FORTH, para carregar o arquivo que contém o módulo principal do sistema GraFORTH. Entretanto, isso é desaconselhável, visto que o DOS 3.3 (ou compatível) normalmente ocupa uma área de memória onde uma parte dela é usada pelo sistema GraFORTH para armazenar o arquivo de trabalho do editor e o editor de texto do GraFORTH.

Se o seu sistema Apple II permitir o boot autostart, basta você desligar o seu Apple II, colocar o disquete com o sistema GraFORTH no drive 1 do slot 6 e ligar o seu Apple II.

Você pode também inicializar o sistema GraFORTH pelo sistema monitor, entrando: C600G, ou teclando: 6⟨CTRL⟩⟨P⟩⟨RETURN⟩.

Os arquivos carregados na inicialização do sistema GraFORTH são:

1.5.0 OBJ.FORTH

Este é o módulo principal do sistema do GraFORTH e responsável pela inicialização. Este programa binário possui rotinas específicas para carregar os editores de textos OBJ.EDITOR1 ou OBJ.EDITOR2 de acordo com o seu sistema Apple II. É neste módulo que estão armazenadas as palavras do sistema.

1.5.1 OBJ.EDITOR1 ou OBJ.EDITOR2

Estes são os arquivos que contêm um dos editores de textos do sistema GraFORTH adequados ao seu sistema Apple II.

O arquivo binário OBJ.EDITOR2 é sempre carregado pelo módulo OBJ.FORTH no seu Apple II desde que seu sistema possua uma placa de expansão de pelo menos 16 Kbytes, totalizando 64 Kbytes de memória total, enquanto que o arquivo binário OBJ.EDITOR1 é carregado para a memória pelo módulo OBJ.FORTH, se seu sistema Apple II tiver somente 48 Kbytes.

1.5.2 CHR.SYS

Este arquivo contém o bloco de caracteres usado pelo sistema GraFORTH, sendo que CHR.SYS é o nome do arquivo do bloco de caracteres requerido pelo módulo OBJ.FORTH.

2

DEMONSTRANDO OS RECURSOS DO GRAFORTH

2.0 Os programas demonstrativos

Após carregar o sistema original do GraFORTH para a memória do seu Apple II, você verá a seguinte mensagem no canto superior esquerdo da tela:

GraFORTH II P. Lutus 1981
Demonstration (Y/N) :

Se você pressionar a tecla ⟨Y⟩, o sistema GraFORTH passará o controle aos programas demonstrativos HEADER e MENU. Entretanto, se você pressionar a tecla ⟨N⟩, ele permitir-lhe-á entrar suas palavras (comandos).

Vamos pressionar a tecla ⟨Y⟩, pois desejamos observar os recursos do sistema GraFORTH.

Respondendo a mensagem com a tecla ⟨Y⟩, o sistema carregará o arquivo de programa HEADER, o qual faz uma demonstração dos recursos de animação (ver a figura 2.0) do sistema GraFORTH.

Pressionando uma tecla qualquer, o programa HEADER compilará e executará o arquivo de textos (programa-fonte) MENU. O programa MENU exibe então uma lista (um menu) dos recursos possíveis no sistema GraFORTH (ver a figura 2.1). E por esse menu, você pode selecionar qualquer um dos programas demonstrativos que deseja executar a fim de conhecer certos recursos do GraFORTH. Acabando a execução de cada um dos programas demonstrativos, o controle volta ao programa MENU.

Para sair do programa MENU, basta selecionar o item "8. Enter GraFORTH II" do menu. E para sair de um dos programas demonstrativos listados no quadro 2.1, basta pressionar as teclas ⟨CTRL⟩⟨RESET⟩.

Os programas demonstrativos do disquete do sistema original do GraFORTH são os arquivos catalogados no Quadro 2.1.

Quadro 2.1 Programas demonstrativos do GraFORTH

```
T 013 QUERY
T 008 HEADER
T 008 MENU
T 008 GRAPHICS1
T 010 GRAPHICS2
T 016 GRAPHICS3
T 012 TEXTDEMO
T 018 FORTHDESC
T 011 FLEDERMAUS
T 011 PIANO
T 004 CLOCK
```

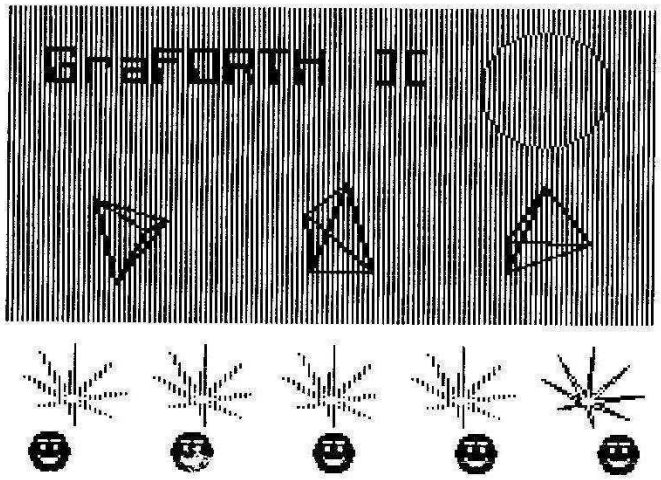


Figura 2.0 – O programa HEADER.


```
GraFORTH JC Demonstrations Menu

GraFORTH JC-(C) P. Lutus 1981

   Here are the demonstration options:

1. Graphics of the First  Kind
2. Graphics of the Second Kind
3. Graphics of the Third  Kind
4. Text Display Options
5. GraFORTH JC System Summary
6. Die Fledermaus (don't miss this!)
7. Music Synthesizer
8. Enter GraFORTH JC

   Enter a number (1-8) : ■
```

Figura 2.1 – O programa MENU.

2.0.0 GRAPHICS1

O programa GRAPHICS1 exibe na tela breves comentários e exemplos de aplicação das palavras gráficas bidimensionais para plotagem de pontos, desenhos de linhas e preenchimento de figuras. Estes recursos são melhor descritos no Capítulo 15.

Selecione o item "1. Graphics of the First Kind" do menu.

2.0.1 GRAPHICS2

O programa GRAPHICS2 demonstra alguns recursos de animação de desenhos bidimensionais através da técnica de operar com blocos de caracteres gráficos. Note que cada figura demonstrada pode ser dividida por caracteres. Ver a parte sexta para maiores detalhes.

Observe bem a mudança dos tamanhos e das cores das figuras durante a demonstração do item "2. Graphics of the Second Kind" do menu.

2.0.2 GRAPHICS3

O GRAPHICS3 é um dos programas demonstrativos mais simpáticos. Ele exibe animações interessantes de diversos objetos tridimensionais. Comenta e mostra os recursos do sistema GraFORTH sobre a rotação dos eixos (X, Y e Z), a translação dos eixos, as escalas e a perspectiva das imagens.

Observe bem a ordem da velocidade das animações demonstradas no item "3. Graphics of the Third Kind" do menu pelo sistema GraFORTH.

Ver também as partes oitava e nona para melhor compreender o uso desses recursos.

2.0.3 TEXTDEMO

O programa TEXTDEMO exibe os vários recursos de tratamento de "textos" pelo sistema GraFORTH. Mostra os vários estilos de grafia, alguns tamanhos possíveis de caracteres, a possibilidade de exibir caracteres nos modos normal e inverso, e o recurso de scrool da tela.

Apresenta ainda vários comentários interessantes e ótimos exemplos ilustrativos.

O TEXTDEMO é executado pelo programa MENU, se selecionarmos o item "4. Text Display Options" do menu dos demonstrativos disponíveis.

Para maiores informações a respeito dos recursos apresentados, ver a parte sexta.

2.0.4 FORTHDESC

O demonstrativo FORTHDESC dá uma breve descrição a respeito do compilador e do editor de textos do GraFORTH, fornecendo alguns comentários interessantes a respeito do binômio edição/compilação.

Selecione o item "5. GraFORTH II System Summary" do MENU para executar este demonstrativo.

2.0.5 FLEDERMAUS

O FLEDERMAUS é o melhor e o mais interessante dos demonstrativos. Este programa exibe o vôo noturno de um morcego, mostrando os recursos de animação dos gráficos tridimensionais do sistema GraFORTH.

Você não deve perder a execução deste programa. Selecione então o item "6. Die Fledermaus (don't miss this!)" do menu.

O FLEDERMAUS será oportunamente melhor analisado no Capítulo 23.

2.0.6 PIANO

Se você gosta de uma boa música, irá apreciar tocar algumas notas no piano exibido por este programa.

É um demonstrativo útil, pelo qual você pode analisar melhor o som e a música gerados pelo sistema GraFORTH.

Execute o demonstrativo PIANO através do programa MENU, selecionando o item "7. Music Synthesizer" do menu.

Para melhor compreender os recursos sonoros do GraFORTH, leia a parte sétima.

2.0.7 QUERY

Este arquivo contém um programa escrito em GraFORTH, quando compilado e executado pelo sistema GraFORTH, exibe a mensagem "Demonstration (Y/N) :" e executa a chamada a rotinas dos programas demonstrativos.

3

COMPONDO UM DISQUETE MAIS ADEQUADO ÀS SUAS APLICAÇÕES

3.0 O disquete do sistema GraFORTH

O disquete original do sistema GraFORTH tem somente cerca de 158 setores livres. Sem dúvida, esta é uma notícia ruim, pois queremos gravar diversos programas em GraFORTH e o espaço disponível de memória nesse disquete é irrisório. Contudo, para a nossa alegria, o Insoft, Inc. possui uma política e uma diretriz de venda e suporte racionais. A empresa autoriza o usuário a fazer cópias (backup copies) do seu sistema original, desde que as cópias sejam exclusivamente para seu uso. E a Insoft, Inc. aconselha também a todos os usuários que não utilizem o disquete original do GraFORTH. Assim, uma vez adquirido o disquete original do sistema GraFORTH, aconselhamos que se trabalhe com disquetes de cópias, poupando o disquete original de um eventual desastre físico ou lógico.

3.1 Fazendo cópias do disquete do GraFORTH

O sistema GraFORTH é acompanhado de um sistema operacional compatível com o DOS 3.3. Você deve ter notado que quando carregamos o sistema GraFORTH, observamos um asterisco (prompt do sistema monitor) no canto inferior esquerdo da tela, ao invés do símbolo] (prompt do Applesoft BASIC). Isso ocorre porque o sistema operacional do GraFORTH carrega um arquivo binário na memória e executa esse programa como sendo seu primeiro programa de apresentação (tal como acontece com o arquivo de programa HELLO em Applesoft BASIC).

A compatibilidade entre ambos os sistemas operacionais é tanta que você pode gravar os seus arquivos de programas-fontes ou de programas-objetos do GraFORTH num disquete com o DOS 3.3 da Apple Computer, Inc. ou compatíveis. Você pode perceber então que fazer cópias do sistema GraFORTH é muito fácil.

Você pode até executar o sistema GraFORTH sob o DOS, desde que não utilize os recursos do editor de textos do sistema GraFORTH

e não desvie o controle do GraFORTH para o sistema operacional.

Você pode usar quaisquer programas de cópias disponíveis no mercado para o Apple II, visto que o sistema GraFORTH não apresenta nenhuma forma de proteção contra cópias.

Uma vez feita a cópia do disquete do sistema GraFORTH, vamos testar esse disquete de cópia.

Coloque o disquete na unidade de disco (disk drive) do seu sistema Apple II e dê a carga (o boot) desse disquete com o sistema GraFORTH.

Se a sua cópia for bem feita, logo você observará as seguintes mensagens:

```
GraFORTH II (C) P. Lutus 1981
Demonstration (Y/N) :
```

Pressione a tecla ⟨Y⟩ para testar todos os recursos do sistema GraFORTH nesse disquete através dos programas demonstrativos disponíveis. Ver o Capítulo 2 para maiores informações sobre esses programas.

3.2 Como selecionar os arquivos

Feita a cópia do disquete original do GraFORTH e a verificação do seu disquete de trabalho, vamos selecionar os arquivos nesse disquete, deletando os arquivos desnecessários às suas aplicações, obtendo assim o espaço disponível para armazenar os nossos programas.

Precisamos então descobrir quais arquivos do sistema são úteis e quais os que não são. Para nosso disquete de trabalho, podemos deletar todos os arquivos de programas demonstrativos.

3.2.0 Selecionando o editor de textos do GraFORTH

Há dois arquivos no sistema GraFORTH, e somente um deles pode estar no seu disquete de trabalho. São os seguintes:

```
B 011 OBJ.EDITOR1
B 011 OBJ.EDITOR2
```

O arquivo OBJ.EDITOR1 deve ser removido e o OBJ.EDITOR2 deve ser mantido, se o seu sistema Apple II possuir apenas 48K bytes de memória, ao passo que, se o seu sistema Apple II possuir as placas de expansão de memória ou Language Card, você deve remover o arquivo OBJ.EDITOR2 e manter intacto o OBJ.EDITOR1.

3.2.1 Uma lista dos arquivos removíveis

Até aqui já apresentamos alguns arquivos que podem ser removidos do seu disquete de trabalho sem afetar o bom funcionamento do sistema GraFORTH.

Uma lista dos arquivos do sistema GraFORTH que podem ser removidos do seu disquete de trabalho sob certas condições de perder ou não determinados recursos do sistema GraFORTH, bem como uma lista dos arquivos irremovíveis, sob pena de perder todos os recursos do GraFORTH ou até o próprio sistema GraFORTH, são apresentadas no Quadro 3.1.

Quadro 3.1 **O diretório do sistema GraFORTH**

OS ARQUIVOS:	CONDIÇÃO:
B 037 OBJ.FORTH	I
B 011 OBJ.EDITOR1	R (se 64K); O (se 48K)
B 011 OBJ.EDITOR2	R (se 48K); O (se 64K)
T 021 CHAREDITOR	O
T 024 IMAGEDITOR	O
T 015 PROFILE	O
T 004 TURTLE	O
T 022 PLAY	O
T 004 STRING WORDS	O
B 005 CHR.SYS	I
B 005 CHR.STOP	O
B 005 CHR.SLANT	O
B 005 CHR.GOTHIC	O
B 005 CHR.BYTE	O
B 005 CHR.STUFF	R
B 005 CHR.MAXWELL	O
T 003 QUERY	O
T 013 HEADER	R
T 008 MENU	R
T 008 GRAPHICS1	R
T 010 GRAPHICS2	R
T 016 GRAPHICS3	R
T 012 TEXTDEMO	R
T 018 FORTHDESC	R
T 011 FLEDERMAUS	R
T 011 PIANO	R
T 004 CLOCK	R
B 002 XYZ	R
B 002 BAT	R

OS ARQUIVOS:	CONDIÇÃO:
B 002 CUBE	R
B 002 HOUSE	R
B 010 CHAL	R
T 003 BIGCHAL	R

Observação:

I – Arquivos irremovíveis. Compromete o funcionamento do GraFORTH.
R – Arquivos removíveis. Não compromete a execução do GraFORTH.
O – Arquivos opcionalmente removíveis, sob comprometimento de certos recursos do GraFORTH.

3.3 Como deletar os arquivos

Uma vez dentro do sistema GraFORTH, você pode deletar os arquivos, passando o controle a partir do GraFORTH para o DOS 3.3 de três modos: pelo sistema GraFORTH, pelo editor de textos do GraFORTH, ou através do Applesoft BASIC.

Vamos descrever minuciosamente esses três processos.

3.3.0 Deletando arquivos no GraFORTH

Se você está no sistema GraFORTH, seu prompt é a palavra Ready, que é exibida no canto esquerdo da tela.

Você pode então utilizar o comando do DOS 3.3 para destravar a proteção dos arquivos (UNLOCK), entrando as seguintes palavras a serem executadas:

```
Ready CR 132 PUTC PRINT " UNLOCK parâmetros-arquivo" CR
```

E para deletar um arquivo, entre as seguintes palavras:

```
Ready CR 132 PUTC PRINT " DELETE parâmetros-arquivo " CR
```

Os parâmetros-arquivo são os exigidos pelo respectivo comando. Ele pode compreender o nome do arquivo, o número do drive, o número do slot e o volume do disquete.

3.3.1 Deletando arquivos no editor do GraFORTH

Se você estiver no sistema GraFORTH, entre a palavra EDIT para chamar o editor de textos do GraFORTH. Dentro do processador, pressione as teclas ⟨CTRL⟩⟨D⟩⟨RETURN⟩ para habilitar a execução dos comandos DOS (ver o Capítulo 10). Você verá a seguinte mensagem na tela:

```
Enter DOS Command :
```

Entre então o comando CATALOG para listar o diretório. Anote os nomes dos arquivos que você deseja apagar, e entre:

```
DELETE parâmetros-arquivo
```

Se o arquivo estiver protegido (locked) contra a deleção, entre, antes, o comando: UNLOCK parâmetros-arquivo.

Para sair desse editor de textos, voltando ao sistema GraFORTH, entre a instrução BYE (ou B) ou pressione as teclas ⟨CTRL⟩⟨RESET⟩.

3.3.2 Deletando arquivos com Applesoft BASIC ativo

Se você deu a carga (o boot) de um disquete com o DOS 3.3 ou compatíveis, você pode listar o diretório de um disquete com o sistema GraFORTH e deletar seus arquivos com o Applesoft BASIC ativo.

Entre os comandos do DOS diretamente no Applesoft BASIC, e o interpretador passará o controle ao DOS.

3.3.3 Deletando arquivos via programa DOS em GraFORTH

No item 13.6 do Capítulo 13, elaboramos um programa utilitário que pode ser usado todas as vezes que se queira passar o controle do GraFORTH para o DOS a fim de executar um comando do sistema operacional.

Para deletar arquivos com esse programa armazenado no disco, entre:

```
Ready READ " DOS "
```

O programa DOS exibirá então a seguinte mensagem no canto superior de sua tela:

```
Entre um Comando DOS:
```

Entre o comando DELETE parâmetros-arquivo, e UNLOCK parâmetros, se necessário.

4

A ESTRUTURA DA LINGUAGEM GRAFORTH

4.0 Uma descrição do sistema GraFORTH

Ao inserir o disquete contendo o sistema GraFORTH na unidade de disco do seu distema Apple II e inicializá-lo, você verá a seguinte mensagem no canto superior esquerdo:

```
GraFORTH II (C) P. Lutus 1981
Demonstration (Y/N) :
```

Se você pressionar a tecla (N) em resposta a essa mensagem, o sistema GraFORTH apagará os textos acima, exibindo os seguintes:

```
GraFORTH II (C) P. Lutus 1981
Ready
```

A mensagem Ready, como você pode perceber, é o prompt do sistema GraFORTH. Isto significa que a mensagem Ready é sempre exibida quando o sistema GraFORTH está pronto para compilar ou executar qualquer instrução que você entrar.

4.1 A linguagem GraFORTH

As instruções da linguagem GraFORTH são compostas por palavras e/ou blocos de palavras. As palavras são as unidades básicas de programação em GraFORTH.

Os dados operados na programação do GraFORTH podem ser lidos ou armazenados na pilha de dados, na pilha de retorno, numa variável numérica, numa variável string ou num arquivo de disco.

Conforme dissemos, a estrutura da linguagem GraFORTH se associa intimamente à programação estruturada. Essa ligação é melhor notada se analisarmos a estrutura da pilha de dados e do dicionário de palavras usadas pela linguagem GraFORTH.

A linguagem GraFORTH é realmente curiosa e interessante. Ela possibilita definir nossas próprias palavras (comandos) dentro do sistema ou esquecer as palavras que não desejamos no dicionário.

O dicionário de palavras é uma parte sempre visível e viva no sistema GraFORTH, onde se armazenam os códigos das palavras do sistema e a tradução dos blocos de palavras dos usuários.

O dicionário de palavras é o conjunto de palavras definidas pelo sistema e que podem ser chamadas e executadas pela simples referência ao seu nome. Por exemplo, a palavra LIST faz parte do dicionário, pois entrando a palavra LIST no sistema GraFORTH, é executada uma determinada função, ou seja, listar todas as palavras do dicionário.

No decorrer das nossas explanações, você entenderá melhor tanto a estrutura da linguagem GraFORTH quanto todos os elementos que fazem parte do sistema GraFORTH.

4.2 As palavras

Conforme já dissemos, ao contrário das outras linguagens de programação que restringem os seus comandos a funções predeterminadas e limitadas, o sistema GraFORTH permite que qualquer usuário possa gerar novas palavras com funções adequadas a suas necessidades, ou seja, as palavras podem ser definidas em função da sua aplicação dentro do seu sistema Apple II na programação.

Percebemos, assim, que as palavras são muito mais do que simples comandos. Elas são as rotinas do usuário, são os procedimentos, são os próprios programas, são as variáveis do sistema e dos programas, enfim toda uma estrutura de uma linguagem bem planejada.

As palavras do GraFORTH podem ser agrupadas em: palavras do sistema e palavras das bibliotecas.

As palavras do sistema são aquelas que são sempre referenciadas pelo GraFORTH, servindo à maioria das aplicações do seu sistema Apple II. São rotinas que vêm gravadas no arquivo OBJ.FORTH.

As palavras das bibliotecas são aquelas criadas no editor de textos ou no compilador, e que não fazem parte do arquivo OBJ.FORTH do sistema. São aquelas que só podem ser referenciadas se compilarmos algum arquivo de programa-fonte. Elas são pois uma extensão do sistema GraFORTH, permitindo gerar novas aplicações no GraFORTH. Um exemplo disso é o arquivo TURTLE que permite desenhar gráficos com palavras em capacidades turtlegráficas.

4.3 Listando as palavras

O sistema GraFORTH permite listar todas as palavras do dicionário num determinado momento. Para tanto, entre:

```
Ready LIST
```

A palavra LIST exibirá então vinte palavras do dicionário de cada vez, e para você ver o resto do dicionário, deverá pressionar qualquer tecla (exceto as teclas ⟨CTRL⟩⟨C⟩ e ⟨CTRL⟩⟨RESET⟩) toda vez que o GraFORTH congelar a listagem. Se você desejar interromper a listagem, pressione as teclas ⟨CTRL⟩⟨C⟩ ou ⟨CTRL⟩⟨RESET⟩.

As teclas ⟨CTRL⟩⟨C⟩ interromperão a listagem das vinte atuais palavras, exibindo, no canto inferior esquerdo, o prompt Ready. As teclas ⟨CTRL⟩⟨RESET⟩ modificam todo o sistema, alterando o atual estado da pilha e do modo gráfico da tela para o estado previamente pré-programado pelo sistema.

Se o seu dicionário não for acrescido de novas palavras ou não for modificado, sua primeira palavra no topo deverá ser CHS.

Observação: a partir deste item, todos os exemplos listados neste livro, que constam do prompt Ready, devem ser entrados diretamente no sistema GraFORTH.

4.4 A pilha de dados

A pilha de dados é uma técnica inteligente de armazenar dados (números) necessários à execução das palavras.

As palavras são executadas conforme a ordem que foram entradas. Os dados numéricos também são armazenados e lidos na ordem em que foram entrados. (Ver o Capítulo 5.) Por exemplo, entre a seguinte rotina:

```
Ready 7 7 + .
```

O sistema GraFORTH colocará os números 7 na pilha e executará a palavra + e depois a palavra . . A palavra + instrui o GraFORTH a retirar os dois últimos números da pilha de dados, somá-los e colocar o resultado no topo da pilha. Enquanto que a palavra . instrui o sistema a retirar o último número da pilha de dados e exibi-lo na tela.

4.5 Os números

Já explicamos que o sistema GraFORTH opera somente com números inteiros na faixa -32768 a +32767.

Contudo, se você entrar números inteiros fora dessa faixa, o sistema GraFORTH converterá o número entrado para um valor dentro da faixa.

Se você entrar um número real maior do que 1 ou menor do que -1 ou se o resultado da operação aritmética não for inteiro, o sistema GraFORTH truncará os números decimais, transformando esses números em valores inteiros. Entre as seguintes palavras e observe o resultado:

```
Ready 7 2 / .
3
```

Os números 7 e 2 são colocados no topo da pilha. A palavra / instrui o sistema GraFORTH a retirar os dois últimos números da pilha e dividir o segundo número retirado pelo primeiro, colocando o resultado no topo da pilha. E a palavra . exibirá, na tela, esse resultado guardado no topo da pilha.

Quando você entra um conjunto de caracteres que se inicia com um ou mais números seguidos de letras ou símbolos especiais, o sistema GraFORTH irá considerar este conjunto de caracteres como sendo a entrada de dados numéricos e colocar no topo da pilha, aliás, o GraFORTH irá somente considerar os números e ignorar os caracteres não-numéricos. Experimente:

```
Ready 777.77 .
777
Ready 123,45 .
123
Ready 723P22312 .
723
```

Entretanto, se você entrar um conjunto de caracteres que se inicia com uma letra alfabética ou um símbolo especial, dependendo da base numérica que você usar e das palavras do dicionário, o sistema GraFORTH exibirá uma mensagem de erro, por exemplo:

```
Ready Z123
Z123 Not Found (Press Return)
```

5

A NOTAÇÃO POLONESA INVERSA

5.0 A notação pós-fixada

A operação matemática em GraFORTH referencia-se à notação pós-fixada (também denominada de notação polonesa inversa, devido ao seu criador, o lógico polonês J. Lukasiewics), a qual convenciona que os operandos devem ser entrados antes do operador. A principal característica dessa notação é que os operadores não se situam entre os operandos, mas sim após estes (notação pós-fixada).

A notação polonesa inversa é uma implementação por demais importante no sistema GraFORTH, pois sem ela não seria possível o GraFORTH trabalhar com a pilha de dados.

Para entender melhor essa notação, vamos compará-la com a notação matemática normal:

NOTAÇÃO NORMAL	NOTAÇÃO POLONESA INVERSA
X + Y	X Y +
X + Y + Z	X Y + Z +
X * (Y + Z)	X Y Z + *
(X OR Y) AND Z	X Y OR Z AND

Como você pode observar, a notação polonesa inversa é, portanto, uma técnica onde as expressões matemáticas e lógicas são representadas por seqüências de operandos e operadores, visando eliminar as dúvidas a respeito das prioridades dos operadores dentro de uma expressão matemática ou lógica, e do uso de parênteses.

5.1 A notação pós-fixada e as palavras

Podemos afirmar que não são somente as palavras matemáticas ou lógicas que obedecem a notação polonesa inversa, mas também todas as outras palavras do sistema GraFORTH que aguardam algum dado na pilha. Observemos que devemos armazenar, na pilha, todos os

dados (operandos) necessários para executar as palavras (operadores). Exemplo:

```
Ready -123 SGN .
```

Como você deve notar, a notação pós-fixada é a notação matemática (ou lógica) usada nas linguagens de máquina e Assembly, visto que você está instruindo o computador a executar as operações no modo definido pelo hardware.

5.2 Uma rotina de conversão de notações matemáticas

Para auxiliar o leitor na conversão da notação matemática normal, com a qual estamos acostumados, para a notação pós-fixada, desenvolvemos uma rotina que converterá qualquer expressão aritmética normal numa expressão polonesa invertida.

A rotina aqui exibida está em Applesoft BASIC. Presumimos que o leitor já possui um bom conhecimento dessa linguagem para entender melhor a lógica que orienta o programa. Se você, entretanto, tiver dúvidas a respeito de alguns dos comandos do Applesoft BASIC, consulte o livro *Guia do programador Applesoft/Integer BASIC*, publicado pela Hemus Editora Ltda.

```
10 REM    ROTINA DE CONVERSAO DE
20 REM    NOTACOES MATEMATICAS
30 REM    VERSAO APPLESOFT BASIC
40 REM
50 DIM PRN$(50),DL$(50)
60 TEXT:HOME:INVERSE:POKE34,0:VTAB1
70 PRINT "ROTINA DE CONVERSAO DE";
80 VTAB2:PRINT "NOTACOES MATEMATICAS"
90 NORMAL:POKE34,5:C%=0:P%=0:D%=0:B$=""
100 INPUT "ENTRE A EXPRESSAO -> ";LINHA$
110 N%=LEN(LINHA$)
120 GOSUB 180:IF A$="(" THEN D%=D%+1:
    DL$(D%)=A$:GOTO 160
130 IF A$=")" THEN GOSUB 250:GOTO 160
140 IF (A$="+")OR(A$="-")OR(A$="*")OR
    (A$="/") THEN GOSUB 290:GOTO 160
150 P%=P%+1:PRN$(P%)=A$:IF ASC(A$)=32
    THEN P%=P%-1
160 IF C%=N% THEN GOTO 390
170 GOTO 120
180 IF C%<N% THEN C%=C%+1:
    A$=MID$(LINHA$,C%,1):GOTO 210
190 IF B$<>"" THEN A$=B$
```

```
200 RETURN
210 IF (A$=)"A") AND (A$(="Z") THEN
    B$=B$+A$:GOTO 180
220 IF (A$=)"0") AND (A$(="9") THEN
    B$=B$+A$:GOTO 180
230 IF B$( )"" THEN C%=C%-1:A$=B$:B$=""
240 RETURN
250 IF D%)0 THEN IF DL$(D%)( )"(" THEN
    P%=P%+1:PRN$(P%)=DL$(D%):D%=D%-1:
    GOTO 250
260 IF D%)0 THEN D%=D%-1
270 RETURN
280 D%=D%+1:DL$(D%)=A$:RETURN
290 IF D%=0 THEN 280
300 IF DL$(D%)="(" THEN 280
310 IF (DL$(D%)="*") OR (DL$(D%)="/")
    THEN P0%=2:GOTO 340
320 IF (DL$(D%)="+") OR (DL$(D%)="-")
 THEN P0%=1:GOTO 340
330 P0%=0
340 IF (A$="*") OR (A$="/") THEN P1%=2:
    GOTO 370
350 IF (A$="+") OR (A$="-") THEN P1%=1:
    GOTO 370
360 P1%=0
370 IF P0%(P1% THEN 280
380 P%=P%+1:PRN$(P%)=DL$(D%):D%=D%-1:
    GOTO 290
390 IF D%)0 THEN P%=P%+1:
    PRN$(P%)=DL$(D%):D%=D%-1:GOTO 390
400 FOR I=0 TO P%:PRINT PRN$(I);" ";:
   NEXT I:PRINT:FLASH
410 PRINT "NOVAMENTE (S/N) : ";:GET A$:
    PRINT A$:IF A$="S" THEN 90
420 SPEED=15:PRINT:PRINT "ADEUS . . .":
    SPEED=255:NORMAL:END
```

Execute este programa no interpretador Applesoft BASIC, entrando uma expressão aritmética de notação matemática normal e ela será transformada numa expressão de notação polonesa inversa pelo programa aqui apresentado.

Como um futuro exercício de programação, deixaremos que o leitor, ao entender a lógica deste programa e ao conhecer melhor a linguagem GraFORTH, venha a implementar esta rotina no GraFORTH.

Parte segunda

Usando o sistema GraFORTH

6

COMO MANIPULAR A PILHA DE DADOS

6.0 A pilha de dados e o GraFORTH

A pilha de dados, usada pelo sistema GraFORTH, é uma técnica de otimizar a estrutura da memória a fim de aumentar a velocidade de processamento e utilizar melhor a memória disponível do seu sistema Apple II.

O sistema GraFORTH permite também o uso das variáveis numéricas para armazenamento de dados. Contudo, você deve saber que o método de acesso aos valores contidos em uma variável numérica é muito mais lento do que o acesso à pilha, e que a variável numérica disputa pela ocupação da memória do seu Apple II para as palavras. Ver o item 8.3 do Capítulo 8.

O método de acesso aos dados numéricos da pilha é o de empilhamento. O método de empilhamento é associado ao conceito LIFO (Last In, First Out). Vamos explicar melhor, ao longo desse capítulo, todos os conceitos aqui descritos.

6.1 Exibindo o conteúdo da pilha de dados

No dicionário do sistema GraFORTH, existe uma palavra que nos exibe o conteúdo da pilha de dados a cada momento do processamento. É uma palavra muito útil na depuração e na confecção dos programas, uma vez que nos permite ver se os valores estão sendo entrados/retirados de acordo com a necessidade do nosso programa. A palavra é STACK. A seguir, descreveremos sua utilização.

Se acabamos de inicializar o sistema GraFORTH, a pilha de dados estará vazia. Senão, entremos a palavra ABORT para simular o estado inicial do sistema GraFORTH. Podemos conferir, usando a palavra STACK.

```
Ready ABORT
Ready STACK
Ready
```

Entraremos então os seguintes números:

```
Ready 1 2 3
[ 1 ]
[ 2 ]
[ 3 ]
Ready
```

Entrando os números 1, 2 e 3, preenchemos a pilha de dados e como a palavra STACK tinha habilitado a exibição constante dos valores da pilha, o sistema GraFORTH exibirá, após cada entrada de dados ou palavras, os valores da pilha.

Os elementos da pilha são exibidos um por linha e limitados pelos colchetes. O topo da pilha é o último elemento exibido e a base da pilha é o primeiro elemento exibido. Como você pode observar, os números são entrados seqüencialmente na pilha. Cada valor entrado ocupava a última posição da pilha, ou seja, cada valor ocupava o topo da pilha no momento em que era entrado. Vamos entrar os seguintes dados para melhor ilustrar essa afirmação:

```
Ready 4
[ 1 ]
[ 2 ]
[ 3 ]
[ 4 ]
Ready 5
[ 1 ]
[ 2 ]
[ 3 ]
[ 4 ]
[ 5 ]
```

Conforme já descrevemos, a operação de leitura/armazenamento de dados na pilha se associa ao conceito LIFO (Last In, First Out), o qual afirma que o último elemento a entrar é o primeiro elemento a sair. Vamos demonstrá-lo com exemplos:

```
Ready . CR .
5
4
[ 1 ]
[ 2 ]
[ 3 ]
Ready
```

A palavra . retira o último elemento do topo da pilha e o exibe

na tela. A palavra CR envia um caractere de controle de linha para o texto, forçando que o próximo caractere ou número a ser exibido seja mostrado na linha seguinte da atual.

Para cancelar o modo habilitado pela palavra STACK, entre a palavra ABORT, a qual restaura as condições iniciais do sistema, ou pressione as teclas ⟨CTRL⟩⟨RESET⟩.

6.2 Manipulando a pilha de dados

A maioria das palavras do GraFORTH manipula os valores contidos na pilha. No Quadro 6.1, podemos distinguir, contudo, cinco palavras essenciais que manipulam os números da pilha.

Quadro 6.1 **As palavras básicas do GraFORTH na manipulação da pilha**

PALAVRA	DESCRIÇÃO
DROP	Retira o último elemento da pilha
DUP	Duplica o elemento do topo da pilha
OVER	Duplica o penúltimo elemento da pilha
PICK	Duplica o num-ésimo elemento da pilha
SWAP	Permuta os últimos elementos da pilha

Essas palavras utilizadas em conjunto possibilitam-nos diversos recursos que nenhuma outra linguagem poderia oferecer. No decorrer deste livro vamos lhe mostrar porque estes recursos são essenciais para a programação em GraFORTH. Analisemos separadamente cada uma dessas palavras que manipulam os elementos da pilha:

6.2.0 A palavra DROP

A palavra DROP retira o último elemento do topo da pilha. Observemos então o efeito dessa palavra sobre a pilha:

```
Ready ABORT
Ready STACK
Ready 11 22 33
[ 11 ]
[ 22 ]
[ 33 ]
Ready DROP
[ 11 ]
[ 22 ]
```

6.2.1 A palavra DUP

A palavra DUP duplica o último elemento da pilha, isto é, ela copia o último valor da pilha, colocando-o no topo da pilha. Por exemplo, entre:

```
Ready ABORT
Ready STACK
Ready 12 24 36
[ 12 ]
[ 24 ]
[ 36 ]
Ready DUP
[ 12 ]
[ 24 ]
[ 36 ]
[ 36 ]
```

6.2.2 A palavra OVER

A palavra OVER duplica o penúltimo elemento da pilha, isto é, ela copia o penúltimo valor da pilha, colocando-o no topo da pilha. Entre as seguintes palavras para ilustrar a manipulação dos dados pela palavra OVER:

```
Ready ABORT
Ready STACK
Ready 64 86
[ 64 ]
[ 86 ]
Ready OVER
[ 64 ]
[ 86 ]
[ 64 ]
```

6.2.3 A palavra PICK

A palavra PICK possui o seguinte formato:

num PICK

Esta palavra aguardará na pilha o valor que indicará a num-ésima posição contendo um elemento a ser duplicado e empilhado no topo da pilha.

Observe o seguinte exemplo de aplicação:

```
Ready ABORT
Ready STACK
Ready 10 20 30 40 3
[ 10 ]
[ 20 ] <-------- posição 3
[ 30 ]
[ 40 ]
[ 3 ]  <-------- posição 0
Ready PICK
[ 10 ]
[ 20 ]
[ 30 ]
[ 40 ]
[ 20 ]
```

O valor 3 indica que a terceira posição a partir dele contém o valor (20) a ser duplicado pela palavra PICK.

6.2.4 A palavra SWAP

Formato: num1 num2 SWAP

A palavra SWAP aguardará na pilha os valores num1 e num2 que serão permutados entre si. Entre as seguintes palavras:

```
Ready ABORT
Ready STACK
Ready 64 12 41
[ 64 ]
[ 12 ]
[ 41 ]
Ready SWAP
[ 64 ]
[ 41 ]
[ 12 ]
```

6.3 As funções matemáticas do GraFORTH

No Quadro 6.2 temos as palavras do dicionário do sistema GraFORTH com funções matemáticas e seus efeitos na pilha, e que iremos analisar.

Quadro 6.2 As principais funções matemáticas do GraFORTH

PALAVRA DO GRAFORTH	DESCRIÇÃO
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
ABS	Função Valor Absoluto
CHS	Função Troca de Sinal
MAX	Função Valor Máximo
MIN	Função Valor Mínimo
MOD	Função Resto da Divisão
RND	Número Randômico
RNDB	Byte Randômico
SGN	Função Sinal
SIN	Função Seno

6.3.0 A adição (+)

Formato: num1 num2 +

A palavra + retira os elementos num1 e num2 da pilha, soma-os e coloca o resultado no topo da pilha.

```
Ready ABORT
Ready STACK
Ready 1 3 5
[ 1 ]
[ 3 ]
[ 5 ]
Ready +
[ 1 ]
[ 8 ]
```

Conforme descrevemos, a palavra + retira o num1 (3) e o num2 (5) previamente colocados na pilha, soma-os (8) e deposita-o no topo da pilha.

6.3.1 A subtração (-)

Formato: num1 num2 -

A palavra - retira os dados num1 e num2 da pilha, subtrai o num2 do num1, isto é, efetua a operação aritmética, num1 - num2, e coloca o resultado no topo da pilha.


```
Ready ABORT
Ready STACK
Ready 7 8 9
[ 7 ]
[ 8 ]
[ 9 ]
Ready -
[ 7 ]
[ -1 ]
Ready -
[ 8 ]
```

6.3.2 A multiplicação (*)

Formato: num1 num2 *

A palavra * retira os dois últimos elementos da pilha, efetua a sua multiplicação, e o resultado (num1 * num2) é colocado no topo da pilha. Entre:

```
Ready ABORT
Ready STACK
Ready 2 -2 3
[ 2 ]
[ -2 ]
[ 3 ]
Ready *
[ 2 ]
[ -6 ]
```

6.3.3 A divisão (/)

Formato: num1 num2 /

A palavra / retira os valores num1 e num2 da pilha, divide o num1 pelo num2 (num1 / num2), e coloca o quociente no topo da pilha.

```
Ready ABORT
Ready STACK
Ready 7 7 2
[ 7 ]
[ 7 ]
[ 2 ]
```

```
Ready /
[ 7 ]
[ 3 ]
Ready .
3
[ 7 ]
```

Observe que a operação 7/2 deveria resultar em um valor real (3,5), mas como o sistema GraFORTH só opera com números inteiros, obteve-se o valor 3. Note também que a palavra . simplesmente retira o valor 3 do topo da pilha e o exibe na tela.

6.3.4 A função valor absoluto (ABS)

Formato: num ABS

A palavra ABS devolve à pilha o valor absoluto (valor sem o sinal) de num.

```
Ready 66 ABS .
66
Ready -77 ABS .
77
```

6.3.5 A função troca de sinal (CHS)

Formato: num CHS

A palavra CHS retira o valor num do topo da pilha, troca o seu sinal e empilha o valor oposto.

```
Ready -123 CHS .
123
Ready 77 CHS .
-77
```

6.3.6 A função valor máximo (MAX)

Formato: num1 num2 MAX

Esta palavra retira os dois últimos elementos da pilha, compara-os, e empilha o maior valor dentre os dois números retirados, ou seja, se num1 > num2, o valor devolvido à pilha será o num1, se não, o num2.


```
Ready 77 35 MAX .
77
Ready 25 86 MAX .
86
```

6.3.7 A função valor mínimo (MIN)

Formato: num1 num2 MIN

A palavra MIN retira o num1 e o num2 previamente empilhados, compara-os, e coloca o menor valor dentre eles no topo da pilha. Se num1 < num2, o valor a empilhar será o num1, se não, será o num2.

```
Ready -12 8 MIN .
-12
Ready -9 -11 MIN .
-11
```

6.3.8 A função resto da divisão (MOD)

Formato: num1 num2 MOD

A palavra MOD retira os últimos elementos da pilha, divide o num1 pelo num2 e empilha o resto dessa divisão (num1/num2).

```
Ready 77 2 MOD .
1
Ready -10 3 MOD .
-1
```

6.3.9 A função valor randômico (RND ou RNDB)

Formato: RND
RNDB

As palavras RND e RNDB geram um número randômico num1 e empilha-o na pilha.

O número num1 gerado pela palavra RND é um valor inteiro menor do que 32767 e maior do que -32767, ou seja, -32767 < num1 < 32767, enquanto que o número num2 gerado pela palavra RNDB é um valor compreendido entre zero e 255, isto é, 0 < num2 < 255.

```
Ready RND .
-5384
Ready RNDB .
227
```

Observe que os números mostrados por essas duas palavras variam a cada entrada e de acordo com as circunstâncias.

6.3.10 A função sinal (SGN)

Formato: num SGN

Essa palavra retira o num da pilha e devolve à pilha o valor 1 se num > 0; o valor 0 se num = 0; ou valor -1 se num < 0.

```
Ready -234 SGN .
-1
Ready 1234 SGN .
1
Ready 0 SGN .
0
```

6.3.11 A função seno (SIN)

Formato: num SIN

A palavra SIN devolve um valor inteiro da função seno de num, onde o valor 128 corresponde a 360 graus, e o valor -128 a -360 graus.

Como o sistema GraFORTH só opera com os números inteiros, esta palavra gerará um valor num1 tal que -128 < num1 < 127.

```
Ready 0 SIN .
0
Ready 32 SIN 127 / .
1
Ready -64 SIN .
0
```

6.4 Manipulando os valores em outras bases numéricas

O sistema GraFORTH nos possibilita operar os números em qualquer base numérica. As palavras que nos permitem selecionar as bases são as seguintes:

BINARY — Com esta palavra, você pode selecionar a base binária para as operações matemáticas.

DECIMAL — A palavra DECIMAL instruirá o sistema GraFORTH a operar com a base decimal que é a base pré-definida.

HEX — Esta palavra instruirá o sistema GraFORTH a ler e exibir os números na base hexadecimal.

BASE — A palavra BASE possibilita-lhe selecionar qualquer base numérica a ser operada. Como esta palavra é na verdade uma variável numérica do sistema, seu formato completo é:

num -) BASE

Vamos demonstrar o uso destas palavras aqui descritas:

```
Ready ABORT
Ready STACK 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 HEX
[ 1 ]
[ 2 ]
[ 3 ]
[ 4 ]
[ 5 ]
[ 6 ]
[ 7 ]
[ 8 ]
[ 9 ]
[ A ]
[ B ]
[ C ]
[ D ]
[ E ]
[ F ]
[10 ]
Ready ABORT
Ready STACK 1 2 3 4 5 6 7 8 BINARY
[ 1 ]
[10 ]
[11 ]
[100 ]
[101 ]
[110 ]
[111 ]
[1000 ]
Ready 111 -) BASE
[ 1 ]
[ 2 ]
[ 3 ]
[ 4 ]
[ 5 ]
```

```
[ 6 ]
[10 ]
[11 ]
Ready DECIMAL
[ 1 ]
[ 2 ]
[ 3 ]
[ 4 ]
[ 5 ]
[ 6 ]
[ 7 ]
[ 8 ]
```

Para a base hexadecimal, os números, ao serem entrados, podem vir precedidos do sinal $.

```
Ready ABORT
Ready HEX $FACA $FE2 STACK
[ FACA ]
[ FE2 ]
```

Você pode entrar um conjunto de caracteres e números compatíveis com a respectiva base numérica. Contudo, se você entrar caracteres incompatíveis após os compatíveis, o sistema GraFORTH irá considerar somente os caracteres compatíveis e ignorar o resto dos caracteres. Se você iniciar um número com caracteres incompatíveis com a base, uma mensagem de erro será exibida:

```
Ready ABORT
Ready STACK HEX
Ready $FAZER
[ FA ]
Ready BINARY
[ 11111010 ]
Ready 212
212 Not Found (Return)
```


7

PREPARANDO O PROGRAMADOR

7.0 Exibindo textos na tela

Há, no sistema GraFORTH, um conjunto de palavras que lhe possibilitam exibir na tela as mensagens que você desejar, tal como acontece em outras linguagens de programação.

Exibir textos é um recurso deveras útil e comum na maioria das linguagens de alto nível, e o sistema GraFORTH não poderia deixar de conter palavras que tratam desse recurso.

7.0.0 Como exibir mensagens na tela

A principal palavra responsável pela manipulação e exibição de textos na tela é PRINT. Seu formato é o seguinte:

```
PRINT " mensagem "
```

A palavra PRINT indicará ao sistema GraFORTH que há uma mensagem a ser exibida na tela. As aspas delimitam o início e o fim de uma mensagem. E como as aspas são palavras no GraFORTH, elas devem conter um espaço separando-as da palavra PRINT e um outro entre as aspas e a mensagem.

Vejamos alguns exemplos:

```
Ready PRINT " GRAFORTH "
GRAFORTH
Ready PRINT " APPLE " PRINT " II "
APPLE II
Ready PRINT " 1  2 " PRINT " 3 "
1  23
Ready PRINT " EXIBINDO "ASPAS" "
EXIBINDO "ASPAS"
```

Como você pode observar no último exemplo, conseguiremos exibir aspas dentro da mensagem desde que elas não sejam isoladas por dois espaços.

No segundo exemplo, podemos notar que a primeira palavra PRINT apenas exibe a mensagem limitada pelas aspas, sem mandar nenhum espaço ou caractere de alimentação de linha no fim da mensagem. Portanto, a segunda palavra PRINT exibiu os caracteres II logo após a mensagem APPLE. Observe bem o terceiro exemplo, pois demonstramos os mesmos efeitos da palavra PRINT.

7.0.1 Como formatar os textos

Já que a palavra PRINT não exibe automaticamente um espaço ou um caractere de alimentação de linha no fim da mensagem mostrada na tela, corremos o risco de exibir várias mensagens uma atrás da outra sem nenhuma formatação. Isso, na verdade, não ocorre graças à existência das palavras SPCE e CR.

A palavra SPCE exibe um espaço na tela, enquanto que a palavra CR exibe um caractere de alimentação de linha. Vejamos seus efeitos junto à palavra PRINT:

```
Ready HOME
      PRINT " EU " SPCE
      PRINT " GOSTO DO " SPCE
      PRINT " APPLE "
EU GOSTO DO APPLE
Ready HOME
      PRINT " EU " CR
      PRINT " GOSTO DO " CR
      PRINT " APPLE "
EU
GOSTO DO
APPLE
```

Como se pode notar nestes exemplos, usamos uma palavra nova: HOME, que limpa toda a tela de textos ou toda a janela de textos no vídeo.

Exibir mensagens na tela é um recurso muito útil na programação, pois elas possibilitam uma melhor interação entre o seu programa e o usuário.

Observação: nas listagens dos programas e/ou dos exemplos, as palavras foram agrupadas em blocos, um em cada linha, visando facilitar a compreensão das palavras entradas. Contudo, elas devem ser entradas uma após outra com um ou mais espaços separando-as.

Há ainda mais três palavras responsáveis pela formatação de textos na tela:

HTAB — É a palavra que indica ao sistema GraFORTH qual é a atual posição horizontal do cursor. A palavra HTAB aguarda no topo da pilha o valor (0 a 39) para a posição atual do cursor.

VTAB — Indica ao sistema qual é a atual posição vertical do cursor. Ela aguarda no topo da pilha o valor (0 a 23) para a tabulação vertical.

WINDOW — Estabelece quais são os limites da janela de textos da tela. Como a janela de textos é uma área retangular, esta palavra aguarda então quatro valores na pilha, os quais indicam os limites da janela.

Vamos demonstrar a aplicação destes recursos entrando o bloco de palavras a seguir. Observe bem os efeitos causados.

```
Ready 30 10 15 22 WINDOW
( DEFININDO UMA JANELA DE TEXTOS )
      HOME
( APAGANDO A TELA )
      PRINT "OS RECURSOS DE FORMATACAO"
      CR
( ALIMENTACAO DE LINHA )
      20 HTAB
( TABULACAO HORIZONTAL )
      PRINT "TABULACAO HORIZONTAL" CR
      22 VTAB PRINT "LINHA 23 DA TELA"
( TABULACAO VERTICAL )
```

Do exemplo acima, podemos concluir que o formato da palavra WINDOW é:

```
num1 num2 num3 num4 WINDOW
```

onde o num1 é o limite direito da janela de textos, o num2 o esquerdo, o num3 o limite superior e o num4 o inferior.

Observação 1: O símbolo (é uma palavra do dicionário que instrui o sistema GraFORTH a ignorar os comentários que a seguem até achar a palavra). O formato das palavras (e) é o seguinte:

```
( comentário )
```

Observação 2: As palavras (e) serão amplamente usadas nas listagens dos programas e dos exemplos. Contudo, você não precisará entrar essas palavras em seus comentários a fim de economizar espaços de memória na edição de programas diretamente no GraFORTH ou no editor de textos do sistema GraFORTH.

7.0.2 Como apagar os textos

Podemos apagar os textos ou limpar a tela no sistema GraFORTH com as seguintes palavras:

HOME Esta palavra apaga a janela de textos na tela.

CLEOP (CLear to End Of Page) apaga os textos desde a atual posição do cursor até o fim da janela de textos.

CLEOL (CLear to End Of Line) apaga os textos desde a atual posição do cursor até o fim da linha.

ERASE Limpa os textos e os gráficos na tela. Esta palavra é executada geralmente mais rápido que a palavra HOME.

Aconselhamos o leitor a encher a tela do vídeo do seu sistema Apple II com os mais variados textos a fim de experimentar as palavras aqui listadas e observar seus efeitos.

7.1 Manipulando os caracteres em código ASCII

Como você sabe, a memória do seu sistema Apple II não armazena caracteres, mas sim números (de 0 a 255). Há duas palavras (PUTC e GETC) no dicionário do sistema GraFORTH que manipulam os caracteres de acordo com o valor do código ASCII apresentado no Quadro 7.1.

Quadro 7.1 Os caracteres e seu valor no código ASCII

VALOR ASCII				
NEGATIVO		POSITIVO		Caractere
Decimal	Hexadecimal	Decimal	Hexadecimal	
128	80	0	0	ConTRoL-@
129	81	1	1	ConTRoL-A
130	82	2	2	ConTRoL-B
131	83	3	3	ConTRoL-C
132	84	4	4	ConTRoL-D
133	85	5	5	ConTRoL-E
134	86	6	6	ConTRoL-F
135	87	7	7	ConTRoL-G
136	88	8	8	ConTRoL-H
137	89	9	9	ConTRoL-I
138	8A	10	A	ConTRoL-J
139	8B	11	B	ConTRoL-K
140	8C	12	C	ConTRoL-L
141	8D	13	D	ConTRoL-M
142	8E	14	E	ConTRoL-N
143	8F	15	F	ConTRoL-O
144	90	16	10	ConTRoL-P
145	91	17	11	ConTRoL-Q
146	92	18	12	ConTRoL-R
147	93	19	13	ConTRoL-S
148	94	20	14	ConTRoL-T
149	95	21	15	ConTRoL-U
150	96	22	16	ConTRoL-V
151	97	23	17	ConTRoL-W
152	98	24	18	ConTRoL-X
153	99	25	19	ConTRoL-Y
154	9A	26	1A	ConTRoL-Z
155	9B	27	1B	ESCape
156	9C	28	1C	\
157	9D	29	1D	]
158	9E	30	1E	^
159	9F	31	1F	nada
160	A0	32	20	espaço
161	A1	33	21	!
162	A2	34	22	"
163	A3	35	23	#
164	A4	36	24	$
165	A5	37	25	%
166	A6	38	26	&
167	A7	39	27	'
168	A8	40	28	(
169	A9	41	29	)
170	AA	42	2A	*
171	AB	43	2B	+

VALOR ASCII				
NEGATIVO		POSITIVO		Caractere
Decimal	Hexadecimal	Decimal	Hexadecimal	
172	AC	44	2C	,
173	AD	45	2D	-
174	AE	46	2E	.
175	AF	47	2F	/
176	B0	48	30	0
177	B1	49	31	1
178	B2	50	32	2
179	B3	51	33	3
180	B4	52	34	4
181	B5	53	35	5
182	B6	54	36	6
183	B7	55	37	7
184	B8	56	38	8
185	B9	57	39	9
186	BA	58	3A	:
187	BB	59	3B	;
188	BC	60	3C	<
189	BD	61	3D	=
190	BE	62	3E	>
191	BF	63	3F	?
192	C0	64	40	@
193	C1	65	41	A
194	C2	66	42	B
195	C3	67	43	C
196	C4	68	44	D
197	C5	69	45	E
198	C6	70	46	F
199	C7	71	47	G
200	C8	72	48	H
201	C9	73	49	I
202	CA	74	4A	J
203	CB	75	4B	K
204	CC	76	4C	L
205	CD	77	4D	M
206	CE	78	4E	N
207	CF	79	4F	O
208	D0	80	50	P
209	D1	81	51	Q
210	D2	82	52	R
211	D3	83	53	S

VALOR ASCII				
NEGATIVO		POSITIVO		Caractere
Decimal	Hexadecimal	Decimal	Hexadecimal	
212	D4	84	54	T
213	D5	85	55	U
214	D6	86	56	V
215	D7	87	57	W
216	D8	88	58	X
217	D9	89	59	Y
218	DA	90	5A	Z
219	DB	91	5B	[
220	DC	92	5C	\
221	DD	93	5D	]
222	DE	94	5E	^
223	DF	95	5F	_
224	E0	96	60	nada
225	E1	97	61	a
226	E2	98	62	b
227	E3	99	63	c
228	E4	100	64	d
229	E5	101	65	e
230	E6	102	66	f
231	E7	103	67	g
232	E8	104	68	h
233	E9	105	69	i
234	EA	106	6A	j
235	EB	107	6B	k
236	EC	108	6C	l
237	ED	109	6D	m
238	EE	110	6E	n
239	EF	111	6F	o
240	F0	112	70	p
241	F1	113	71	q
242	F2	114	72	r
243	F3	115	73	s
244	F4	116	74	t
245	F5	117	75	u
246	F6	118	76	v
247	F7	119	77	w
248	F8	120	78	x
249	F9	121	79	y
250	FA	122	7A	z

Você deve utilizar os "números negativos" do quadro, no sistema GraFORTH, para os valores dos caracteres em código ASCII.

7.1.0 A palavra PUTC

A palavra PUTC do sistema GraFORTH permite-lhe exibir um caractere a partir de um valor armazenado no topo da pilha. Ela exibe um caractere na tela de acordo com o valor ASCII do número dado. O formato da palavra PUTC é:

```
Ready num PUTC
```

Portanto, a palavra PUTC (PUT Character) retira um elemento (num) do topo da pilha, interpreta o número segundo o código ASCII para um caractere e exibe esse caractere na atual posição do cursor. Por exemplo:

```
Ready 199 PUTC  242 PUTC  225 PUTC  198 PUTC
      207 PUTC  210 PUTC  212 PUTC  200 PUTC
GraFORTH
Ready 135 PUTC ( tocando campainha )
```

7.1.1 A palavra GETC

A palavra GETC (GET Character) do dicionário do sistema instrui o GraFORTH a exibir o cursor na atual posição da tela, a esperar pelo acionamento de uma tecla qualquer, e a empilhar o valor em código ASCII da tecla pressionada. Entre:

```
Ready PRINT " PRESSIONE UMA TECLA "
      SPCE GETC DUP CR
      PRINT " CODIGO ASCII " SPCE . CR
      PRINT " CARACTER " PUTC
```

Neste exemplo, você percebe então que a palavra GETC obtém o valor do caractere digitado a partir do teclado em código ASCII e coloca-o no topo da pilha, e para exibir o caractere teclado deve-se usar a palavra PUTC.

7.2 Entrando os programas diretamente no sistema GraFORTH

Podemos entrar diretamente os nossos programas no sistema GraFORTH, o qual compilará todo um bloco de palavras e executá-lo-á

imediatamente ou não, de acordo com as instruções de entradas. Assim, não podemos editar (modificar) o bloco de palavras entrado diretamente no GraFORTH, pois o compilador do sistema GraFORTH não deixa nenhum vestígio do programa-fonte.

Talvez você já deve ter percebido que o sistema GraFORTH exibe os seus textos e executa seus programas no modo gráfico de alta resolução do sistema Apple II. O sistema GraFORTH pode tanto exibir os textos como também os gráficos. Discutiremos isso no Capítulo XIV. Para a entrada de blocos de palavras e programas no GraFORTH, esse sistema simulou algumas técnicas de edição tais como as existentes nos interpretadores Applesoft/Integer BASIC. Encontramos pois, no sistema GraFORTH, os mais variados caracteres de controle intimamente ligados às técnicas de edição do sistema GraFORTH, as quais são bem mais visíveis no editor de textos do GraFORTH.

Vamos analisar os caracteres de controle fornecidos pelas teclas utilizadas conjuntamente com a tecla ⟨CTRL⟩ (ou ⟨CONTROL⟩) ou com a tecla ⟨ESC⟩ (ou ⟨ESCAPE⟩), de uso especial pelo sistema GraFORTH. Esses caracteres possuem muitas utilidades na entrada de um programa e na sua edição.

7.2.1 Os caracteres de controle de letras maiúsculas/minúsculas

Os caracteres, aqui apresentados, controlam a entrada e a saída das letras, habilitando os caracteres de maiúsculas ou minúsculas para entrar e exibir as letras, ou exibindo símbolos especiais. Veja o Quadro 7.2.

Quadro 7.2 Os caracteres de controle I

⟨CTRL⟩⟨O⟩⟨E⟩ habilita entrar dados e editar textos com os caracteres de minúsculas. Pode-se entrar os caracteres de maiúsculas se a tecla ⟨ESC⟩ for acionada antes de digitar a letra.
Experimente:

```
Ready  ( ⟨CTRL⟩⟨O⟩⟨E⟩ letras minusculas )
Ready  ( ⟨ESC⟩Maiusculas ⟨ESC⟩M )
```

⟨CTRL⟩⟨O⟩⟨U⟩ habilita entrar dados e editar textos com caracteres de maiúsculas somente, dependendo do seu sistema Apple II.
Digite:

```
Ready  ( ⟨CTRL⟩⟨O⟩⟨E⟩minusculas )
```

Ready (⟨CTRL⟩⟨O⟩⟨U⟩MAIUSCULAS)

⟨CTRL⟩⟨O⟩⟨L⟩	habilita entrar dados e editar textos com caracteres de minúsculas somente, dependendo do seu sistema Apple II. Digite o seguinte exemplo:

Ready (⟨CTRL⟩⟨O⟩⟨L⟩ver os exemplos anteriores)

⟨CTRL⟩⟨SHIFT⟩⟨M⟩	exibe o símbolo _ (sublinhar), dependendo do seu sistema Apple II.
⟨CTRL⟩⟨SHIFT⟩⟨N⟩	exibe o símbolo [(abrir colchete), dependendo do seu sistema Apple II.
⟨SHIFT⟩⟨M⟩	exibe o símbolo] (fechar colchete).
⟨CTRL⟩⟨G⟩	toca um som de campainha.

7.2.2 Os caracteres de edição

Os caracteres de edição do sistema GraFORTH e do seu editor de textos são apresentados no Quadro 7.3.

Quadro 7.3 **Os caracteres de controle II**

⟨CTRL⟩⟨H⟩	move o cursor um caractere à esquerda e apaga o caractere anteriormente entrado. Corresponde à tecla (<—).
⟨CTRL⟩⟨I⟩	insere um espaço branco entre duas letras.
⟨CTRL⟩⟨K⟩	move o cursor um caractere para cima. Se o cursor estiver na primeira linha da tela, os textos da tela descem uma linha na tela na mesma coluna.
⟨CTRL⟩⟨L⟩	limpa a tela toda (corresponde a executar a palavra ERASE).
⟨CTRL⟩⟨M⟩	corresponde à tecla ⟨RETURN⟩.
⟨CTRL⟩⟨U⟩	move o cursor um caractere à direita e entrando o caractere anterior. Corresponde à tecla (—>).
⟨ESC⟩⟨I⟩	move o cursor um caractere para cima.
⟨ESC⟩⟨J⟩	move o cursor um caractere para a esquerda.
⟨ESC⟩⟨K⟩	move o cursor um caractere para a direita.
⟨ESC⟩⟨M⟩	move o cursor um caractere para baixo.

Observação: Os caracteres de controle que usam a tecla ⟨ESC⟩ só funcionam com o modo habilitado pelas teclas ⟨CTRL⟩⟨O⟩⟨U⟩, o modo pré-definido pelo sistema onde há entrada/saída de letras e com caracteres de maiúsculas.

7.3 Usando as teclas de edição

Vamos demonstrar agora o uso das teclas de edição. Digite as seguintes letras sem pressionar a tecla ⟨RETURN⟩ no fim da linha:

```
Ready PRIT " TESTE "
```

Como você deve ter notado no exemplo dado, faltou a letra N na palavra PRINT. Pressione o cursor (<—) ou as teclas ⟨CTRL⟩⟨H⟩ até a letra I:

```
Ready PRIT " TESTE "
```

Acione as teclas ⟨CTRL⟩⟨I⟩ e o sistema providenciará um espaço entre as letras I e T:

```
Ready PRI T " TESTE "
```

Digite a letra N:

```
Ready PRINT " TESTE "
```

Pressione então a tecla (—>) ou as teclas ⟨CTRL⟩⟨U⟩ até o fim da linha para reentrar os caracteres à direita do N:

```
Ready PRINT " TESTE "
```

Você pode também usar as teclas de edição que só funcionam se a tecla ⟨ESC⟩ for acionada. Por exemplo, entre a seguinte instrução, operando uma mensagem de erro:

```
Ready PRIT " TESTE " ( Pressione ⟨RETURN⟩ )
PRIT Not Found (Return)
Ready
```

Após a exibição da mensagem de erro, pressione a tecla ⟨ESC⟩ para habilitar o modo de edição. Acione então as teclas I, J, K ou M para posicionar o cursor (veja a figura 7.3).

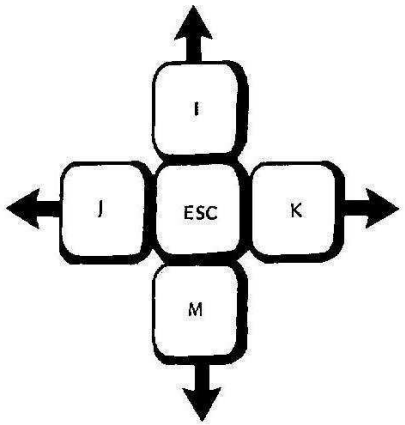


Figura 7.3 – As teclas de edição.

Neste exemplo, posicione o cursor na primeira letra da palavra PRIT, pressione a tecla (–>) para reentrar os caracteres PRI, e acione as teclas ⟨CTRL⟩⟨I⟩ para inserir um espaço entre a letra I e T:

```
PRI T
```

Digite a letra N e reentre o resto da linha:

```
PRINT " TESTE "
```

Podemos então pressionar a tecla ⟨RETURN⟩, ou entrar no modo de edição, pressionando a tecla ⟨ESC⟩ para reposicionar o cursor.

8

PREPARANDO O PROGRAMADOR – II

8.0 Definindo novas palavras no GraFORTH

A capacidade de aplicações do sistema GraFORTH é quase ilimitada devido à possibilidade de introdução de novas palavras dentro do dicionário do GraFORTH, as quais podem ser constantemente esquecidas, isto é, essas novas palavras podem somente existir no momento em que você delas necessitar para determinadas aplicações e depois ser retiradas do dicionário de palavras. Como você pode notar, toda a estrutura de um programa na linguagem GraFORTH gira em torno da constante manipulação do bloco de palavras, sendo que as novas palavras implicam a chamada das já existentes no dicionário.

Vamos definir uma nova palavra:

```
Ready : SOMA
Ready   PRINT " ESTA PALAVRA EXIBE "
Ready   SPCE
Ready   PRINT " A SOMA DE 2 NUMEROS "
Ready   CR
Ready   OVER OVER SWAP
Ready   PRINT " ( " . PRINT " ) + ( "
Ready   . PRINT " ) = " SPCE + . ;
```

Entrando o programa SOMA, você notará que ao entrar a palavra :, o sistema GraFORTH apenas montará o bloco de palavras da estrutura da definição da nova palavra SOMA sem executar nenhuma das palavras entradas após o símbolo : até que a palavra ; seja encontrada. Você também pode entrar toda a definição de uma nova palavra numa só linha.

Se você entrou o programa anterior, criou uma nova palavra denominada SOMA. Podemos verificar se ela está ou não no dicionário de palavras, executando a palavra LIST.

O formato da instrução para definir uma nova palavra é o seguinte:

```
: nome-da-palavra    bloco-de-palavras  ;
```

onde o nome-da-palavra é o nome da nova palavra a ser definida no dicionário, o bloco-de-palavras é o bloco de palavras a ser executado pela nova palavra; a palavra : (dois pontos) indica o início da definição de uma nova palavra, e a palavra ; (ponto-e-vírgula) indica o fim da definição da nova palavra.

Podemos então executar a palavra SOMA já definida no sistema, entrando:

```
Ready -78 77 SOMA
ESTA PALAVRA EXIBE A SOMA DE 2 NUMEROS
(-78) + (77) = -1
```

Dentro de um bloco-de-palavras também é possível aparecer o nome da palavra que está sendo definida, de modo que ela possa chamar a si mesma durante a execução. Este recurso é conhecido como recursividade.

Vejamos um exemplo de uso de recursividade, criando uma palavra que calcula o fatorial de um número:

```
Ready : FAT
Ready   DUP
Ready   Ø=1F
Ready   DROP 1
Ready   ELSE
Ready   DUP 1 - FAT *
Ready   THEN ;
Ready   5 FAT .
120
```

8.1 Esquecendo palavras

Podemos, depois de ter definido a palavra SOMA, esquecê-la:

```
Ready FORGET SOMA
```

A palavra FORGET serve para retirarmos as palavras não usadas do dicionário, obtendo assim uma boa economia de espaço de memória. Com o decorrer do curso, vamos explicar a importância da palavra FORGET no sistema GraFORTH.

Execute a palavra LIST antes e depois de esquecer a palavra.

Se você tentar esquecer uma palavra que não existe no dicionário, o sistema GraFORTH não exibirá nenhuma mensagem de erro.

Se você tentar esquecer uma palavra que não se situa no topo do dicionário, você acabará por apagar todas as palavras do dicionário desde a palavra que você realmente quer esquecer à palavra do topo do dicionário. Exemplificando:

```
Ready : IT ( NAO FAZ NADA ) ;
```



```
Ready : list LIST ABORT ;
Ready : SOMA + . ;
Ready : PILHA STACK ;
Ready LIST
PILHA
SOMA
list
IT
CHS
ABS
( etc . . . )
Ready FORGET IT
CHS
ABS
( etc . . . )
```

Se você não desejar ficar listando o dicionário de palavras do sistema GraFORTH quando precisar apagar as novas palavras que foram acrescidas, defina uma palavra IT (tal como o exemplo dado) e entre a instrução FORGET IT todas as vezes que desejar deletar todas as palavras montadas no dicionário.

8.2 Manipulando entradas de dados

Tentamos demonstrar-lhe a utilidade da pilha de dados e das palavras básicas de manipulação da pilha na entrada de valores com o programa exemplo SOMA. Vamos então analisar melhor esta nova palavra:

```
Ready : SOMA
        PRINT " ESTA PALAVRA EXIBE "
        SPCE
        PRINT " A SOMA DE 2 NUMEROS "
        CR
        OVER OVER SWAP
        PRINT " ( " . PRINT " ) + ( "
        . PRINT " ) = " SPCE
        + . ;
```

A primeira linha (: SOMA) faz parte da estrutura de definição de palavras.

A segunda, a terceira, a quarta e a quinta linha exibem uma mensagem na janela de textos da tela.

Na sexta linha, temos as palavras básicas de manipulação de elementos da pilha. Se a pilha continha os dois últimos elementos assim dispostos:

[num1]
[num2]

com a execução das duas palavras OVER, a pilha passa a ter os seguintes elementos:

[num1]
[num2]
[num1]
[num2]

A palavra SWAP permuta os dois últimos elementos da pilha:

[num1]
[num2]
[num2]
[num1]

A sétima linha exibe (num1), e a suposta linha passa a conter os elementos:

[num1]
[num2]
[num2]

A oitava linha exibe (num2), e o estado da suposta pilha passa a ser:

[num1]
[num2]

Como você pode notar, as palavras OVER serviram para exibir os valores num1 e num2 sem modificar o estado da pilha. Você pode estudar melhor os efeitos causados pela execução do bloco de palavras definido na palavra SOMA, entrando os dados num1 e num2, e as palavras uma por uma, com a exibição do estado da pilha habilitada pela palavra STACK.

Vamos aqui somente descrever o estado da pilha conforme a entrada das principais palavras contidas na SOMA:

```
Ready ABORT
Ready STACK
Ready 99 11
[99 ]
[11 ]
Ready OVER
[99 ]
[11 ]
```



```
[99 ]
Ready OVER
[99 ]
[11 ]
[99 ]
[11 ]
Ready SWAP
[99 ]
[11 ]
[11 ]
[99 ]
Ready . SPCE .
99 11
[99 ]
[11 ]
Ready + .
110
```

A manipulação dos elementos da pilha torna a execução do programa muito mais rápida, pois, ao contrário do método de acesso às variáveis, não requer um tempo extra pra ler/armazenar os dados, fazendo pesquisas no dicionário de palavras a fim de achar as variáveis.

8.3 Definindo as variáveis numéricas

O sistema GraFORTH possui uma palavra, VARIABLE, que permite criar novas palavras, as quais servem apenas para armazenamento de valores numéricos. Ela tem os seguintes formatos:

```
VARIABLE nome-variável
num VARIABLE nome-variável
```

onde o nome-variável é o nome da palavra a ser definida na instrução VARIABLE como uma variável e o num é o valor a ser armazenado inicialmente no nome-variável.

O primeiro formato inicializa a palavra nome-variável como sendo uma variável com valor zero. Enquanto que o segundo formato define o nome-variável com o valor num dado.

Execute a palavra LIST antes e depois de entrar o seguinte exemplo:

```
Ready VARIABLE EXEMPLO
```

Ao executar a palavra LIST logo depois de definir uma variável numérica, podemos observar que o nome dessa variável é a primeira palavra do dicionário.

As variáveis numéricas, por serem palavras, podem ser esquecidas.

```
Ready FORGET NUM1
Ready FORGET NUM2
Ready FORGET EXEMPLO
```

Vamos demonstrar o uso da variável numérica com o seguinte programa:

```
Ready VARIABLE NUM1
Ready 77 VARIABLE NUM2
Ready : SOMA2
        PRINT " ( " NUM1 . PRINT " ) "
        SPCE PRINT " + ( " NUM2 .
        PRINT " ) = " SPCE
        NUM1 NUM2 / . ;
Ready SOMA2
(0) + (77) = 77
Ready 66 -> NUM1
Ready -86 -> NUM2
Ready SOMA2
(66) + (-86) = -20
```

Podemos, assim, concluir que as variáveis numéricas são úteis nos programas para armazenar um valor freqüentemente referenciado e que permanece quase imutável durante o processamento. As variáveis numéricas são um bom recurso para o programador iniciante em GraFORTH, visto que programar com a pilha de dados é um pouco complicado para os usuários menos experientes.

Para verificar o valor contido dentro de uma variável numérica, basta entrar o nome-variável e o sistema se encarregará de colocar o valor da variável no topo da pilha. E para designarmos um valor a uma variável numérica, usamos a palavra –>, cujo formato é:

num –> nome-variável

A palavra –> aguardará na pilha o valor a designar a variável numérica específica na instrução.

A palavra VARIABLE apresenta algumas inconveniências que podem ser evitadas, se seguirmos estes conselhos:

1) a linha do programa que conter a palavra VARIABLE deverá conter somente esta instrução. Senão:

```
Ready ABORT
Ready STACK 77
[ 77 ]
```



```
Ready 12 1 + 86 VARIABLE VALOR
[ 77 ]
Ready VALOR .
VALOR Not Found (Return)
```

2) a palavra VARIABLE não deve ser usada dentro de uma instrução que define uma nova palavra, isto é, não deve vir entre as palavras : e ;. Senão:

```
Ready ABORT
Ready STACK 77
[ 77 ]
Ready : TESTE VARIABLE NUM NUM . ;
NUM Not Found (Return)
```

8.4 O processo de compilação

Todas as palavras entradas via teclado no sistema GraFORTH são imediatamente traduzidas e executadas. O sistema GraFORTH também possibilita-lhe compilar os arquivos gerados pelo editor de textos do GraFORTH ou por outros processadores de textos, ou o conteúdo de uma variável string do dicionário de palavras.

A princípio, devíamos apresentar-lhe os recursos e aplicações do editor de textos e das variáveis strings. Contudo, à medida que você ler este livro, irá conhecê-los.

O sistema GraFORTH permite compilar arquivos de programas-fontes a partir do disco ou da memória do seu Apple II. Existem duas palavras no dicionário, responsáveis por estes dois processos de compilação: READ e MEMRD.

No decorrer das nossas explanações sobre o sistema GraFORTH, as aplicações possíveis destas duas palavras serão fartamente descritas.

Observação: devido aos diversos recursos do GraFORTH, tais como desenho e animação de gráficos tridimensionais, ou comandos TURTLEGRAPHICS, ocuparem uma boa parte de memória do seu sistema Apple II, você deverá, sempre que quiser compilar um arquivo de programa-fonte, esquecer as palavras adicionais criadas pela compilação do arquivo anterior para não ter problemas de memória insuficiente.

8.4.0 A palavra READ

A palavra READ é responsável pela compilação dos programas-fontes armazenados num arquivo de textos do disco. Seu formato é:

READ " nome-programa "

onde o nome-programa é o nome do arquivo do programa-fonte.

A palavra READ instruirá o sistema GraFORTH para compilar todas as instruções do programa-fonte até o fim do arquivo, ou até encontrar um erro de sintaxe. Todas as instruções envolvendo as palavras já montadas no dicionário são executadas seqüencialmente. Portanto, as definições de novas palavras no programa-fonte são acrescidas no dicionário e podem ser executadas imediatamente ou não pelo GraFORTH. Se o programa-fonte apenas foi compilado, podemos executar o programa-objeto, entrando a palavra principal de controle do programa, que é geralmente a primeira do topo do dicionário de palavras. Por exemplo, compilemos e executemos o programa-fonte QUERY:

```
Ready READ " QUERY "
Ready IT
```

A palavra READ compila o programa QUERY armazenado em disco. Entrando a palavra LIST, você percebe que a primeira palavra do dicionário é IT. Entre então a palavra IT para instruir o sistema GraFORTH a executar essa palavra criada na compilação.

8.4.1 A palavra MEMRD

Usa-se a palavra MEMRD do dicionário para instruir o sistema GraFORTH a compilar e executar um arquivo de programa-fonte armazenado na memória do seu sistema Apple II. Seu formato é:

```
endereço-inicial  MEMRD
```

onde o endereço-inicial é o endereço da memória do seu sistema Apple II, a partir do qual o arquivo de programa-fonte está armazenado.

Nos próximos capítulos vamos demonstrar e explicar melhor o uso da palavra MEMRD no sistema GraFORTH.

8.5 As palavras de controle de execução

Há, no dicionário do sistema GraFORTH, duas palavras responsáveis (RUN e AUTORUN) pelo controle de execução de programas no GraFORTH.

8.5.0 A palavra RUN

A palavra RUN instrui o sistema GraFORTH a executar a primeira palavra no topo do dicionário. Esta palavra é útil na medida em que, quando se compila um arquivo de programa-fonte, o nome da rotina principal do programa seja geralmente a última palavra a ser empilhada no dicionário. Por exemplo:

```
Ready READ " QUERY "
Ready RUN
```

8.5.1 A palavra AUTORUN

A palavra AUTORUN habilita o sistema GraFORTH a executar automaticamente a primeira palavra do topo do dicionário. Para habilitar ou desativar o modo AUTORUN, entre a palavra AUTORUN no formato:

```
num  AUTORUN
```

onde o num é um valor não-nulo, se deseja habilitar o modo AUTORUN; ou nulo se deseja desativar o modo AUTORUN já habilitado.

Se o modo AUTORUN for habilitado, então todas às vezes que o controle voltar ao sistema GraFORTH, será executada a primeira palavra que estiver no topo do dicionário, seja ela qual for. Assim, os erros de execução e de sintaxe, a execução da palavra ABORT, ou o acionamento das teclas ⟨CTRL⟩⟨RESET⟩ farão com que o sistema GraFORTH execute a primeira palavra do dicionário. Entre a seguinte demonstração:

```
Ready ABORT
Ready : TESTE
          PRINT " AUTORUN ATIVO!!! " ;
Ready 1 AUTORUN
AUTORUN ATIVO!!!
Ready STACK 1 2
AUTORUN ATIVO!!!
[1]
[2]
Ready ABORT ( REINICIALIZA O SISTEMA )

GraFORTH II (C) P. Lutus 1981
AUTORUN ATIVO!!!
Ready ( PRESSIONE ⟨CTRL⟩⟨RESET⟩ )
       ( REINICIALIZANDO O SISTEMA )
```

```
GraFORTH II (C) P. Lutus 1981
AUTORUN ATIVO!!!
Ready 0 AUTORUN ( DESATIVA AUTORUN )
```

Como você pode notar, para desativar o modo AUTORUN, basta entrar:

```
Ready 0 AUTORUN
```

8.6 Salvando o sistema GraFORTH

A palavra SAVEPRG do dicionário do sistema GraFORTH nos possibilita salvar todo o GraFORTH com o dicionário de palavras modificado, criando um arquivo binário tal como "OBJ.FORTH" ou armazenando no próprio arquivo "OBJ.FORTH". E uma vez que esse arquivo esteja armazenado num disquete, podemos executá-lo via DOS ou dentro do sistema GraFORTH. A palavra SAVEPRG é, assim, uma ferramenta versátil e poderosa.

Conforme mencionamos no Capítulo 3, a Insoft, Inc. nos autoriza a fazer cópias do seu sistema GraFORTH para nosso uso pessoal. Podemos então salvar num disquete um arquivo binário "OBJ. FORTH" modificado, o qual será o mais adequado às nossas aplicações.

Inicialmente, compomos o sistema GraFORTH de acordo com as nossas necessidades (aqui, vamos preparar um arquivo "OBJ.FORTH" que não executará a rotina do programa QUERY). Só depois disso executamos então a palavra SAVEPRG:

```
Ready ABORT
Ready LIST
Ready SAVEPRG
```

Observamos inicialmente quais as palavras do atual dicionário que nos interessam, executando a palavra LIST. Se há palavras inúteis, esqueçam-nas (FORGET). Feito isso, entremos a palavra SAVEPRG, a qual irá exibir a mensagem:

```
Save File Name :
```

Entre o nome do seu arquivo. Neste caso, vamos entrar:

```
OBJ.FORTH
```

Feito isso, uma segunda mensagem é exibida na tela, perguntando se você quer ou não habilitar o modo AUTORUN:

```
Autorun (Y/N) :
```


Neste exemplo, responderemos ⟨N⟩, e o sistema logo providenciará a gravação do atual sistema GraFORTH no arquivo "OBJ.FORTH".

Vamos assim carregar e executar o arquivo binário "OBJ.FORTH" aqui modificado, através do editor de textos do sistema GraFORTH:

```
Ready EDIT
```

Entrando a palavra EDIT, o sistema apagará a tela e exibirá a mensagem:

```
GraFORTH II Editor (C) P. Lutus 1981
```

Pressionando as teclas ⟨CTRL⟩⟨D⟩⟨RETURN⟩, uma mensagem é logo exibida, pedindo que entre um comando do DOS. Entre:

```
Enter DOS Command : BRUN OBJ.FORTH
```

Você irá ver que o novo sistema GraFORTH não exibirá mais a mensagem "Demonstration (Y/N) :".

Ao longo deste livro, vamos usar e abusar da palavra SAVEPRG. Portanto, reflita e tente compreender melhor esta palavra antes de prosseguir a leitura.

8.7 Saindo do sistema GraFORTH

Se você quiser sair do sistema GraFORTH e executar um outro software no seu sistema Apple II, você pode simplesmente desligar o seu sistema computador e ligá-lo novamente com um novo disquete no drive 1 do slot 6. Contudo, há várias outras maneiras inteligentes e sutis para sair do GraFORTH a fim de entrar outros sistemas.

8.7.0 Carregando outros sistemas através das palavras

Uma das formas de sair do GraFORTH e carregar um outro sistema é pela execução de determinadas palavras para passar o controle do GraFORTH para o sistema operacional. Podemos instruir o sistema GraFORTH a carregar um novo disco, entrando:

```
Ready CR 132 PUTC    PRINT " IN#6 " CR
```

ou

```
Ready : PR#6 CR 132 PUTC    PRINT " PR#6 " CR ;
Ready PR#6
```

8.7.1 Carregando outros sistemas através do editor do GraFORTH

Uma outra maneira de carregar um novo sistema é pelo editor de textos do sistema GraFORTH. Entre as palavras EDIT e o sistema GraFORTH executará o seu editor de textos. Acionando as teclas ⟨CTRL⟩⟨D⟩⟨RETURN⟩, o editor exibirá a mensagem "Enter DOS Command :" que aguardará sua instrução:

```
Enter DOS Command : IN#6
```

8.7.2 Carregando outros sistemas através do programa DOS

Pode-se também executar o arquivo de programa-fonte DOS fornecido no item 13.6 do Capítulo 13, entrando:

```
Ready READ " DOS "
```

A seguinte mensagem é exibida. Responda com o comando IN#6:
Entre um comando DOS : IN#6.

8.7.3 Carregando outros sistemas através do sistema monitor

Há uma palavra no dicionário do GraFORTH, a qual faz com que o sistema GraFORTH seja desativado e o controle passe ao sistema monitor do seu Apple II:

```
Ready BYE
*
```

Entrando a palavra BYE, o sistema GraFORTH é então desativado e o prompt do sistema monitor (*) é exibido no canto inferior esquerdo. Você pode, assim, entrar a instrução "C600G" ou pressionar as teclas ⟨6⟩⟨CTRL⟩⟨P⟩⟨RETURN⟩ no sistema monitor para carregar um novo sistema no drive 1 do slot 6. Dentro do sistema monitor, se você entrar a instrução "6000G", o controle voltará ao sistema GraFORTH.

Parte terceira

O editor de textos do sistema GraFORTH

9

COMO EDITAR PROGRAMAS NO EDITOR DE TEXTOS

9.0 Apresentando o editor de textos

Conforme mencionamos no item 3.2.1 do Capítulo 3, o sistema GraFORTH dispõe de dois arquivos binários contendo os editores de textos: "OBJ.EDITOR1" e "OBJ.EDITOR2".

O primeiro é usado por sistemas Apple II sem nenhuma placa de expansão de memória ou Language Card. Possibilita a edição de cerca de 2.045 caracteres (bytes) no seu arquivo de trabalho (arquivo de textos armazenado na memória).

O segundo é carregado se o seu sistema Apple II conter somente uma placa de expansão de memória ou Language Card. Permite a edição de cerca de 11.773 caracteres no seu arquivo de trabalho.

Fora as diferenças aqui citadas, os dois editores de textos do sistema GraFORTH possuem as mesmas características.

Você deve ter notado que os arquivos de programas-fontes do GraFORTH são arquivos de textos do sistema operacional DOS. Você pode, portanto, se não gostar do editor de textos do sistema GraFORTH, usar "qualquer" um outro editor de textos disponível para Apple II em DOS.

Esses dois editores trabalham com uma faixa de memória do sistema Apple II própria para o armazenamento dos dados do editor. Essa faixa de memória é aqui denominada de arquivo de trabalho.

9.1 Entrando no editor de textos

Para executar o editor de textos do sistema GraFORTH, basta entrar a palavra EDIT:

```
Ready EDIT
```

Se seu arquivo binário contendo o respectivo editor de textos não estiver carregado na memória, o sistema GraFORTH providenciará a carga deste na memória. Em poucos segundos, a tela é apagada e o

sistema GraFORTH exibe então a seguinte mensagem no canto superior esquerdo da tela:

GraFORTH II Editor (C) P. Lutus 1981

O prompt no editor de textos é o próprio cursor piscando.

Uma vez que as novas palavras acrescidas no dicionário do GraFORTH podem invadir o espaço de memória disponível para o arquivo de trabalho do editor de textos, aconselhamos que você esqueça (FORGET) as novas palavras do dicionário e apague todo o arquivo de trabalho antes de iniciar a edição/entrada de textos ou programas-fontes. E se você precisar das novas palavras, montadas no dicionário, pode salvar (SAVEPRG) todo o dicionário nesse instante, antes de esquecê-las e entrar a palavra EDIT.

Para facilitar a edição dos seus textos ou programas, o editor possui os comandos mostrados ao lado no Quadro 9.1.

9.2 Editando programas e textos

No editor de textos do sistema GraFORTH, para entrar um bloco de textos ou palavras, cada bloco deve ser precedido de um número. Esse número indica ao editor uma linha específica do arquivo de trabalho. Uma vez que o editor enumera os blocos de textos, fica mais fácil corrigir e inserir textos entre os blocos.

Portanto, se você estiver no editor de textos do sistema GraFORTH poderá entrar as seguintes instruções:

```
VARIABLE NUM
: DOS
  CR 132 PUTC  PRINT ;
: CATALOG
  DOS " CATALOG " ;
: IN#6
  PRINT " CARREGANDO UM NOVO DISQUETE "
  CR
  DOS " IN#6 " ;
: PRINCIPAL
( TESTE DO EDITOR DE TEXTOS )
PRINT " CATALOG E IN#6 DISPONIVEIS " CR
7 -> NUM
NUM . ;
```

Os programas deste livro, que devem ser entrados pelo editor, poderão ser listados dessa maneira e não constarão do prompt "Ready".

Quadro 9.1 Comandos do editor de textos do GraFORTH

LETRA	COMANDO	DESCRIÇÃO
A	AUTONUM	Numera e entra automaticamente as linhas dos programas ou textos.
B	BYE	Desativa o editor de textos, retornando o controle ao sistema GraFORTH.
D	DELETE	Elimina as linhas especificadas dentro do arquivo de trabalho.
E	ERASE	Apaga o arquivo de trabalho.
G	GET	Copia o conteúdo do arquivo de disco para o arquivo de trabalho.
I	INSERT	Posiciona a linha a inserir no seu arquivo de trabalho.
L	LIST	Lista o conteúdo do arquivo de trabalho com pausas a cada 16 linhas de textos.
M	MEMORY	Exibe o número de bytes disponíveis para a edição de textos.
P	PROGRAM	Exibe o endereço inicial do arquivo de trabalho, permitindo sua modificação.
S	SAVE	Salva o arquivo de trabalho no disquete.
W	WRITE	Lista o conteúdo de todo o arquivo de trabalho sem pausas.
⟨CTRL⟩⟨D⟩⟨RETURN⟩		Permite entrada dos comandos do DOS.

Para entrar as instruções apresentadas acima, proceda da seguinte forma:

```
5  VARIABLE NUM
15 : DOS CR 132 PUTC  PRINT ;
25 : CATALOG DOS " CATALOG " ;
35 : IN#6
45 PRINT " CARREGANDO UM NOVO DISQUETE "
```

```
55 CR
65 DOS " IN#6 " ;
75 : PRINCIPAL
85 ( TESTE DO EDITOR DE PROGRAMAS )
95 PRINT " CATALOG E IN#6 DISPONIVEIS "
105 CR
115 7 -> NUM NUM . ;
```

Como você pode notar, as palavras devem ser agrupadas em blocos. Entramos então cada bloco de palavras precedido de um número tal como ocorre em BASIC. Portanto, cada linha deve conter um bloco de palavras composto de acordo com a sintaxe das palavras disponíveis. Por exemplo, a instrução VARIABLE deve ser única numa linha do programa (ver a linha de número 5).

Apresentada a estrutura da entrada de textos no editor do sistema GraFORTH, vamos analisar os seus comandos listados no Quadro 9.1.

9.3 Listando o arquivo de trabalho

Se você entrou todas as linhas do programa-exemplo do item 9.2, você pode listar o arquivo de trabalho, entrando o caractere L ou o comando LIST:

```
L  ( LIST )
10 VARIABLE NUM
20 : DOS CR 132 PUTC  PRINT ;
30 : CATALOG DOS " CATALOG " ;
40 : IN#6
50 PRINT " CARREGANDO UM NOVO DISQUETE "
60 CR
70 DOS " IN#6 " ;
80 : PRINCIPAL
90 ( TESTE DO EDITOR DE PROGRAMAS )
100 PRINT " CATALOG E IN#6 DISPONIVEIS "
110 CR
120 7 -> NUM NUM . ;
Done
```

O comando LIST do editor lista todo o arquivo de trabalho (ou uma parte dele), com pausas a cada 16 linhas de textos. Se você acionar as teclas ⟨CTRL⟩⟨C⟩ durante a pausa, a listagem é logo interrompida. Acabada ou interrompida a listagem pelo comando LIST, o editor exibirá a mensagem **Done**, indicando o fim da execução do comando LIST.

Insistimos para que você entre os blocos das palavras, iniciando a numeração das linhas com o valor 5 e incrementando os números das linhas de 10 em 10. Isso objetiva demonstrar a renumeração automática das linhas pelo editor. O editor de textos do sistema GraFORTH renumera as linhas do arquivo de textos a cada entrada/exibição dos blocos de textos, de modo que a numeração das linhas do arquivo de trabalho se inicie com o valor 10 e tenha incrementos de 10 em 10.

Para listar determinadas linhas ou faixas de linhas do arquivo de trabalho, você deve usar o comando LIST nos seguintes formatos, especificando o intervalo de linhas do arquivo de trabalho:

```
L 70      ( LISTA SOMENTE A LINHA 70 )
L 30,60   ( LISTA DA LINHA 30 ATE A LINHA 60 )
L ,20     ( LISTA DO COMECO DO ARQUIVO ATE A LINHA 20 )
L 40,     ( LISTA DA LINHA 40 ATE O FIM DO ARQUIVO )
```

Você pode usar também o comando WRITE para listar todo o conteúdo do arquivo de trabalho sem pausas. O comando WRITE tem o uso mais adequado na exibição de textos pela impressora, enquanto que o comando LIST deve ser usado na exibição de textos na tela do seu vídeo.

9.4 Deletando linhas no arquivo de trabalho

No editor de textos do sistema GraFORTH, há o comando DELETE que apaga uma faixa de linhas ou uma linha do arquivo de trabalho. Seu formato é idêntico ao do comando LIST:

```
D 10      ( DELETA SOMENTE A LINHA 10 )
D 40,50   ( DELETA DA LINHA 40 ATE A LINHA 50 )
D ,20     ( DELETA DO COMECO DO ARQUIVO ATE A
            LINHA 20 )
D 60,     ( DELETA DA LINHA 60 ATE O FIM DO ARQUIVO )
```

Você pode referenciar uma faixa de linhas inexistente ou a uma linha inexistente, pois o editor não irá modificar nada no seu arquivo de trabalho.

Visto que a enumeração das linhas é sempre automaticamente redefinida a cada entrada/exibição dos blocos de textos, você não deve esquecer que a numeração das linhas anteriores à execução do comando DELETE pode ser diferente da numeração posterior. Suponha a seguinte titulação do arquivo de trabalho:

```
L  ( LIST )
10 : IN#6 CR 132 PUTC  PRINT " IN#6 " CR ;
```

```
20 : PRINCIPAL
30 HOME
40 PRINT " IN#6 PRESENTE NO DICIONARIO "
50 ABORT ;
60 ( AQUI ERA LINHA 60 )
Done
```

Vamos deletar as linhas 20, 30, 40 e 50, e então listaremos novamente este arquivo de trabalho:

```
D 20,50  ( DELETE 20,50 )
Done
L  ( LIST )
10 : IN#6 CR 132 PUTC  PRINT " IN#6 " CR ;
20 ( AQUI ERA LINHA 60 )
Done
```

9.5 Deletando todo o arquivo de trabalho

Para deletar todo o arquivo de trabalho, poderíamos usar o comando DELETE para apagar desde a primeira linha do arquivo de trabalho até a última. Contudo, podemos entrar o comando ERASE ou a letra E para obter o mesmo efeito. O editor exibirá a seguinte mensagem, perguntando-lhe se você realmente quer apagar o arquivo em memória:

```
Erase (Y/N) :
```

Entre Y, se você realmente quer apagar o arquivo de trabalho, se não, entre N.

9.6 O modo AUTONUM

O modo AUTONUM do editor de textos do sistema GraFORTH gera a enumeração automática de linhas para você entrar seus blocos de palavras ou textos. Entre o comando AUTONUM ou a letra A para habilitar este modo.

```
A  ( AUTONUM )
10
```

Entrando a letra A ou o comando AUTONUM, o editor exibirá um número no canto esquerdo e o cursor após este número é o

prompt, indicando que você deve digitar um bloco de textos nessa linha e pressionar a tecla ⟨RETURN⟩ no final dos seus textos:

```
10 : list LIST  ABORT ;
```

O editor então exibirá o número da próxima linha, que será sempre o anterior mais 10:

```
20
```

Para sair do modo AUTONUM, basta você não acionar nenhuma outra tecla após a exibição do número além da tecla ⟨RETURN⟩.

9.7 Inserindo linhas dentro do arquivo de trabalho

Suponhamos que o seu arquivo de trabalho tenha as seguintes linhas exibidas pelo comando LIST do editor de textos do sistema GraFORTH:

```
L  ( LIST )
10 ( PROGRAMA TESTE )
20 PRINT " EXIBINDO O VALOR X1 " CR
30 ( X1 E UMA VARIAVEL )
40 PRINT " DANCOU . . . " CR
50 77 -> X1
60 X1 CHS . ;
Done
```

Como você pode notar, o programa será compilado com erros, não podendo ser executado, pois a variável X1 não foi definida e falta uma parte da instrução para definir uma nova palavra, pois há uma palavra ; no fim do programa. Você precisa corrigir esses erros no editor de textos do sistema GraFORTH antes da compilação. Precisamos então inserir os seguintes blocos de palavras entre as linhas 10 e 20:

```
VARIABLE X1
:  TESTE
```

Podemos entrar os referidos blocos desta maneira:

```
12 VARIABLE X1
14 : TESTE
```

Ou então usar o comando INSERT juntamente com o comando AUTONUM:

```
I  15 ( INSERT 15 )
A     ( AUTONUM )
20
```

Como o comando INSERT deve ser utilizado para reposicionar a inserção de linhas dentro de um arquivo de trabalho, entramos a letra I (ou o comando INSERT) e a linha 15, indicando ao editor que queremos inserir algumas linhas no programa depois da linha 10 e antes da linha 20. Concluímos este modo de inserção automático entrando a letra A ou o comando AUTONUM para entrada de blocos de palavras. O editor exibe então o número da linha 20. Entre, assim, os blocos das palavras desejados e liste o conteúdo do arquivo de trabalho após terminar a inserção:

```
I  15 ( INSERT 15 )
A     ( AUTONUM )
20 VARIABLE X1
30 : TESTE
40

L  ( LIST )
10 ( PROGRAMA TESTE )
20 VARIABLE X1
30 : TESTE
40 PRINT " EXIBINDO O VALOR X1 " CR
50 ( X1 E UMA VARIAVEL )
60 PRINT " DANCOU . . . " CR
70 77 -> X1
80 X1 CHS . ;
Done
```

Para reposicionar a inserção de linhas no arquivo de trabalho depois da última linha do programa, podemos executar o comando INSERT para redefinir a posição da inserção de linha ou qualquer outro comando do editor. Analisemos a atual situação: acabamos de inserir as linhas 20 e 30, e executamos o comando LIST. Se ativarmos o modo AUTONUM novamente, podemos verificar que a posição para inserção de linhas é depois da última linha do programa:

```
A ( AUTONUM )
90
```


10

MANIPULANDO AS ENTRADAS/SAÍDAS COM O EDITOR

10.0 Salvando seu arquivo de trabalho

O objetivo principal do editor de textos é possibilitar salvar o seu programa-fonte ou seus textos editados ou uma parte deles num arquivo de texto de um disquete. Entre a letra S ou o comando SAVE para salvar o seu arquivo de trabalho no disquete:

```
S  ( SAVE )
(Filename) :
```

Executando o comando SAVE, o editor exibirá a mensagem " (Filename) :". Entre então o nome do seu arquivo de textos e, se desejar, pode especificar também os números do slot, drive e volume do disquete, tais como são usados no formato-padrão do DOS. Por exemplo, entre o seguinte nome para salvar todo o arquivo de trabalho:

```
(Filename) : TESTE.DISCO,S6,D1
```

O comando SAVE possibilita também salvar somente determinadas linhas do arquivo de trabalho, bastando entrar o símbolo / e a faixa de linhas do programa que você quer salvar. Experimente um desses comandos:

```
(Filename) : TESTE.DISCO/20,50
      ( SALVA AS LINHAS DE 20 ATE 50 )
(Filename) : TESTE.DISCO/,40
      ( SALVA O ARQUIVO ATE A LINHA 40 )
(Filename) : TESTE.DISCO/70,
      ( SALVA AS LINHAS DE 70 EM DIANTE )
(Filename) : TESTE.DISCO/70
      ( SALVA SOMENTE A LINHA 70 )
```

Você pode deduzir então que o símbolo / não pode ser usado no nome do seu arquivo de textos no disco.

10.1 Carregando um arquivo de textos

Podemos carregar um arquivo de textos do disquete para a memória, inserindo os textos para dentro do arquivo de trabalho, entrando o comando GET ou a letra G:

```
G (GET)
(Filename) :
```

Após entrar o comando GET, a mensagem " (Filename) :" é exibida pelo editor. Entre então o nome do seu arquivo e, se for necessário, especifique também os números do slot, drive e volume do disquete. Por exemplo, entre:

```
(Filename) : TESTE.DISCO
```

Entrando o nome do arquivo no formato acima, estamos inserindo o conteúdo do arquivo TESTE.DISCO do disco após a última linha do arquivo de trabalho antes da carga de leitura.

Podemos também inserir o conteúdo de um arquivo do disco, numa posição específica dentro do arquivo de trabalho. O símbolo / é um valor após o nome do arquivo a indicar que o arquivo a carregar deve ser inserido na linha citada. Experimente:

```
(Filename) : TESTE.DISCO/15
```

A resposta "TESTE.DISCO/15" indica que o nome do arquivo é "TESTE.DISCO" e o arquivo lido deve ser inserido entre as linhas 10 e 20 do arquivo do trabalho.

Como você pode deduzir, não se pode usar o símbolo / no nome do seu arquivo de textos no disco, visto que tanto o comando SAVE quanto o comando GET o utilizam como um parâmetro.

Você deve ter notado que o comando GET nunca apaga o conteúdo do seu arquivo de trabalho, pois ele sempre insere o conteúdo do arquivo do disquete no arquivo de trabalho. Portanto, entre o comando LIST ou ERASE, antes de executar o comando GET.

Tanto o comando GET quanto o comando SAVE, quando entrados, podem exibir a mensagem " (Filename) :" junto com o nome do último arquivo do disco carregado ou armazenado. Você pode percorrer este nome com a tecla (—>) ou as teclas ⟨CTRL⟩⟨U⟩ sem precisar digitá-lo novamente, se desejar utilizá-lo.

10.2 Executando os comandos do sistema operacional via editor

Para executar os comandos do sistema operacional do GraFORTH dentro do seu editor de textos, basta pressionar as teclas ⟨CTRL⟩⟨D⟩

⟨RETURN⟩, e a seguinte mensagem é exibida na tela do seu sistema Apple II:

```
Enter DOS Command :
```

Entre então qualquer um dos comandos do DOS que você desejar, tais como para listar o diretório (CATALOG), apagar um arquivo qualquer no disco (DELETE), proteger (LOCK) um arquivo contra a deleção, destravar a proteção de um arquivo (UNLOCK), carregar um arquivo binário (BLOAD) etc. Você pode executar diversos comandos do DOS, pois este comando só será cancelado após pressionar a tecla ⟨RETURN⟩ duas vezes seguidas, sem entrar nenhum comando do DOS ou entrar alguma instrução errada ou incompatível.

10.3 Listando os textos na impressora

Para imprimir o conteúdo arquivado de trabalho no editor de textos do sistema GraFORTH, você deve ativar a impressora a princípio. Dentro do editor de textos, você deve passar o controle do editor para o sistema operacional, digitando as teclas ⟨CTRL⟩⟨D⟩⟨RETURN⟩ e entrando então o comando PR#1 (desde que a interface da sua impressora esteja conectada no slot 1; senão, substitua o valor 1 pelo número do slot, onde está conectada a interface da impressora):

```
Enter DOS Command : PR#1
```

Com a saída à impressora habilitada, pressione a tecla ⟨RETURN⟩ duas vezes seguidas para devolver o controle ao editor do GraFORTH a partir do sistema operacional.

Entre o comando WRITE ou a letra W para listar todo o conteúdo do arquivo de trabalho sem pausas ou execute o comando LIST.

Para desconectar a saída à impressora, você deverá pressionar as teclas ⟨CTRL⟩⟨RESET⟩, visto que o comando PR#0 não fará com que o sistema GraFORTH mostre os seus textos no modo gráfico de alta resolução.

10.4 Verificando a memória disponível para seu arquivo de trabalho

Para verificar o número de caracteres (bytes) que o editor ainda lhe permite entrar no arquivo de trabalho, entre o comando MEMORY ou a letra M para verificar o espaço de memória disponível no arquivo de trabalho do editor de textos.

Se o seu sistema Apple II tiver somente 48 Kbytes de memória,

entrando o comando MEMORY com todo o conteúdo do arquivo de trabalho deletado você verá a seguinte mensagem:

```
Free Memory 2045
```

No entanto, se o seu sistema Apple II tiver placas de expansão de memória ou Language Card, entrando o comando MEMORY após apagar todo o conteúdo do arquivo de trabalho, você verá a seguinte mensagem:

```
Free Memory 11773
```

Os números podem variar, pois se você não deletou o arquivo de trabalho, o comando MEMORY exibe os mais variados números de acordo com a quantidade de caracteres já entrados no arquivo de trabalho.

10.5 Como modificar o espaço de memória para seu arquivo de trabalho

Você pode achar o espaço de memória disponível para o arquivo de trabalho muito pequeno para entrar/editar o seu programa ou os seus textos. Você deseja então aumentar o espaço de memória, pode querer diminuir o espaço de memória disponível para o seu arquivo de trabalho, em vez de esquecer (FORGET) as novas palavras incluídas no dicionário do sistema.

O sistema GraFORTH possibilita realizar este seu desejo, reajustando o ponteiro do sistema para o limite inferior do arquivo de trabalho. Entre então o comando PROGRAM ou a letra P, e a seguinte mensagem será exibida na tela:

```
E  ( ERASE )
Erase (Y/N) : Y
Done

P  ( PROGRAM )
Program Lenght 0
Position 34817
Free Memory 11773
Change Position (Y/N) :
```

Essa mensagem exibe informações importantes sobre a posição (endereço) inicial e o comprimento do seu arquivo de trabalho, além do espaço de memória ainda disponível, e lhe pergunta se você quer ou não modificar o limite inferior do arquivo de trabalho.

Entre Y, se você desejar modificar o limite inferior do seu arquivo de trabalho, se não, entre N.

Se você entrar Y, o editor exibirá a seguinte mensagem:

```
"ENTER NEW POSITION : "
```

Você pode então entrar o novo endereço inicial do arquivo de trabalho de acordo com as suas aplicações.

10.6 Ampliando o espaço de memória para seu arquivo de trabalho

Se o seu sistema Apple II não possuir nenhuma placa de expansão de memória ou Language Card, o espaço de memória disponível para você editar os seus textos (ou programas-fontes) será irrisório (2.045 caracteres); você vai desejar, assim, aumentar o espaço de memória do seu arquivo de trabalho. Se seu sistema Apple II possuir alguma placa de expansão ou Language Card e você desejar editar um arquivo de trabalho com mais de 11.773 caracteres, você vai também querer aumentar o espaço de memória do seu arquivo de trabalho.

Podemos resolver esses problemas usando um outro editor de textos disponível para os sistemas Apple II. Do contrário, teremos de mudar a área de memória disponível para o arquivo de trabalho; visto que, se aumentarmos o espaço de memória da atual área, invadiremos a área de memória reservada para o dicionário de palavras do sistema GraFORTH, logo abaixo da atual área de memória disponível para o seu arquivo de trabalho.

Uma área de memória razoavelmente grande, que podemos redefinir como o nosso arquivo de trabalho, é a área de memória reservada para as duas páginas de alta resolução do sistema Apple II. Porém, as duas páginas de alta resolução são usadas pelo sistema GraFORTH para exibir seus textos e gráficos. Teremos então de instruir o sistema GraFORTH a utilizar a primeira página de textos/baixa resolução para exibir seus textos e deixar livres as de alta resolução.

Proceda, portanto, da seguinte maneira a partir do sistema GraFORTH:

```
Ready HOME TEXT
```

Entrando a palavra TEXT, que instruirá o sistema GraFORTH a habilitar a primeira página de textos, será exibida na tela o prompt Ready com os seus caracteres em letras maiúsculas. Execute então a palavra EDIT:

```
Ready EDIT
```

Dentro do editor de textos do sistema GraFORTH, apague o seu arquivo de trabalho e entre a letra P ou o comando PROGRAM, o qual deve exibir a seguinte mensagem:

```
E  ( ERASE )
ERASE (Y/N) : Y
DONE

P  ( PROGRAM )
PROGRAM LENGHT 0
POSITION 34817
FREE MEMORY 11773
CHANGE POSITION (Y/N) :
```

Se o seu sistema Apple II não tiver nenhuma placa de expansão de memória ou Language Card, em vez de 11.773 espaços disponíveis de memória, teríamos 2.045.

Entrando a letra Y, aparecerá a mensagem "ENTER NEW POSITION :" na tela. Entre o número 8191. Delete todo o arquivo de trabalho aqui definido para evitar eventuais sujeiras dessa área de memória. Podemos, assim, entrar novamente o comando PROGRAM no editor para verificar a modificação efetuada:

```
P  ( PROGRAM )
PROGRAM LENGHT 0
POSITION 8192
FREE MEMORY 38398
CHANGE POSITION (Y/N) :
```

Como você deve saber, o valor decimal 8192 corresponde ao valor hexadecimal $2000. Portanto, a posição 8192 da memória é o endereço inicial da primeira página de alta resolução. O número de caracteres que você pode utilizar nesse novo arquivo de trabalho não é 38.398 como mostra a mensagem exibida pelo comando PROGRAM, mas sim, somente o espaço de memória compreendido pelas duas páginas de alta resolução (16.384 bytes ou caracteres), pois acima das duas páginas de alta resolução fica a área de memória reservada ao dicionário de palavras do sistema GraFORTH, a qual não pode ser utilizada pelo editor.

Observe o mapa de memória do sistema GraFORTH no Apêndice para maiores detalhes.

Se você usar as duas páginas de alta resolução do sistema Apple II para armazenar seu programa na memória, você deve salvar o seu arquivo de trabalho sempre que sair do editor de textos do GraFORTH, pois o sistema GraFORTH sempre apaga as duas páginas de alta resolução ao sair do editor de textos.

11

O EDITOR E O SISTEMA GRAFORTH

11.0 Compilando arquivos do editor de textos

Devemos sempre entrar programas no editor de textos do sistema GraFORTH, visto que nele podemos salvar os programas-fontes editados num arquivo de textos do disco para posterior correção ou modificação.

Uma vez editado um arquivo de programa-fonte pelo editor, podemos compilá-lo a partir do disco, entrando a palavra READ no sistema GraFORTH ou executando a palavra MEMRD para compilar o arquivo de trabalho na memória do seu Apple II.

11.0.0 Compilando arquivos do disco

No seu arquivo de programa-fonte armazenado no disco, você pode compilar e executar esse programa, entrando apenas a palavra READ.

Se o seu arquivo de textos possuir somente definições de novas palavras, você compilará e executará o seu arquivo, desde que no fim desse arquivo constem as instruções CLOSE RUN; as quais instruirão o GraFORTH a executar a última palavra nova empilhada no dicionário do sistema logo que o compilar.

Entre as seguintes palavras no editor e salve o arquivo de trabalho num arquivo do disco, denominando-o de "TESTE.COMP":

```
( ARQUIVO : TESTE.COMP )
FORGET IT
: IT ;
: IN#6
  CR 132 PUTC  PRINT " IN#6 " CR ;
: list
  LIST ABORT ;
: BASE7
  7 -> BASE ;
```

```
: PRINCIPAL
  PRINT " list IN#6 e BASE7 " CR
  PRINT " DISPONIVEIS NO DICIONARIO " CR
  list ;
CLOSE RUN
```

Se você entrar as palavras CLOSE RUN no fim do seu arquivo de programa-fonte, notará que quando o sistema GraFORTH encontrar as palavras CLOSE RUN executará a última palavra empilhada no dicionário. E se você não entrar as palavras CLOSE RUN no programa, o sistema GraFORTH apenas compilará esse arquivo de textos do disco ao executar a palavra READ e executará a única instrução direta (FORGET IT); você terá de entrar a palavra RUN ou PRINCIPAL (definida no programa "TESTE.COMP") para executar esse programa.

O programa-exemplo "TESTE.COMP" se inicia com a instrução FORGET IT. Vamos definir a palavra IT em todos os nossos arquivos de programas-fontes como a primeira nova palavra a ser incluída no dicionário do sistema GraFORTH e para apagar todas as novas palavras definidas antes de compilar todos os nossos programas, vamos executar somente esquecer a palavra IT sem precisar listar a toda hora o dicionário de palavras para saber qual delas devemos esquecer.

O recurso de iniciar todos os programas com a instrução FORGET IT e finalizá-los com a instrução CLOSE RUN é muito útil. Analisemos seus benefícios:

```
Ready READ " TESTE.COMP "
list IN#6 e BASE7
DISPONIVEIS NO DICIONARIO
PRINCIPAL
BASE7
list
IN#6
IT
CHS
SGN
(etc.)
```

Executando a instrução READ " TESTE.COMP", observamos que a palavra IT é realmente a primeira palavra desse programa acrescida ao dicionário do sistema GraFORTH.

Vamos então compilar novamente o arquivo "TESTE.COMP" sem apagar essas palavras.


```
Ready READ " TESTE.COMP "
list IN#6 e BASE 7
DISPONIVEIS NO DICIONARIO
PRINCIPAL
BASE7
list
IN#6
IT
CHS
SGN
(etc.)
```

Experimente agora entrar no editor de textos do sistema GraFORTH, apagar as instruções FORGET IT ou as instruções CLOSE RUN, salvá-lo num arquivo de disco e compilá-lo duas vezes seguidas e você poderá deparar com alguma mensagem de erro.

11.0.1 Compilando o arquivo de trabalho

Você pode também compilar o arquivo de trabalho diretamente na memória do sistema Apple II, entrando o endereço inicial do arquivo de trabalho junto com a palavra MEMRD.

O endereço inicial do arquivo de trabalho é geralmente 34817 na base decimal. Se você modificou esse endereço com o comando PROGRAM do editor, basta você entrar o comando PROGRAM no editor de textos do sistema GraFORTH e ver o número que segue a mensagem POSITION para saber qual é o novo endereço.

Portanto, se você acabou de editar um programa pelo editor de textos do sistema GraFORTH, você pode compilar esse programa armazenado no arquivo de trabalho, entrando a seguinte instrução:

```
Ready 34817 MEMRD
```

Observe bem a execução da palavra MEMRD, pois ela compila o programa do arquivo de trabalho e executa-o logo após a compilação, tal como aconteceria na compilação dos arquivos de textos em disco pela palavra READ.

Se você armazenar o programa do arquivo "TESTE.COMP" no arquivo de trabalho sem a instrução CLOSE RUN, a palavra MEMRD compilará o arquivo de trabalho e não executará a palavra PRINCIPAL, visto que a palavra : instrui o sistema GraFORTH a empilhar apenas uma nova palavra no dicionário. Não esqueça que as instruções diretas, tais como FORGET IT, serão executadas seqüencialmente na compilação.

11.1 Como chamar o editor na carga do sistema GraFORTH

Esqueça inicialmente todas as novas palavras empilhadas no dicionário do sistema GraFORTH. Então, só depois disso, defina a seguinte palavra:

```
Ready : IT
        0 AUTORUN       ( desliga o modo AUTORUN )
        EDIT ;          ( carrega/habilita o editor )
```

Com a palavra IT assim definida no topo do dicionário, execute a palavra SAVEPRG para salvar o sistema GraFORTH tal como está:

```
Ready SAVEPRG
Save File Name : OBJ.FORTH
Autorun (Y/N) : Y
```

Agora recarregue o seu sistema GraFORTH. Se você entrou, de maneira correta, todas as instruções aqui fornecidas, o sistema GraFORTH carregará o editor de textos logo que o módulo OBJ.FORTH for carregado. Se ocorrer algum erro, releia o item 8.6 do Capítulo 8.

Parte quarta

Programando em GraFORTH

12

ESTRUTURA DE PROGRAMAÇÃO EM GRAFORTH

12.0 Introduzindo as estruturas de programação do GraFORTH

Conforme já mencionamos, o sistema GraFORTH possui uma estrutura de programação intimamente ligada à programação estruturada. Você pode observar esta filosofia de programação através da estrutura das palavras. Na figura 12.0 você observa melhor a estrutura do dicionário de palavras do sistema GraFORTH. Vamos agora confirmar essa filosofia do GraFORTH através das suas palavras condicionais de execução e loopings (laços).

```
: list
  LIST
  ABORT ;
: PRINCIPAL
  HOME
  list ;
```

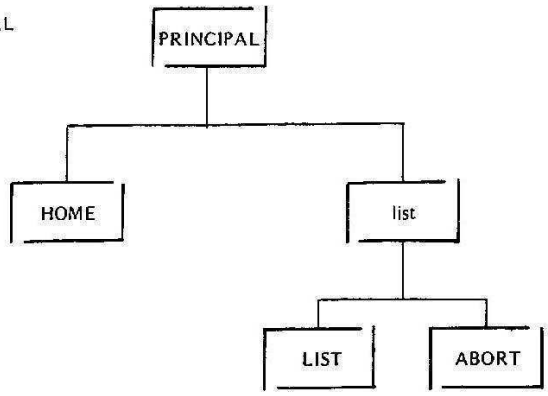


Figura 12.0 — A estrutura das palavras do GraFORTH.

Neste capítulo, desenvolvemos todo um trabalho visando acostumar o leitor à programação estruturada, uma técnica muito usada pela maioria dos programadores profissionais. Não dispensamos nenhuma forma e/ou técnica de explanação possível para explicar todas as palavras condicionais de execução e laços do sistema GraFORTH.

As estruturas condicionais de enlaces (loopings) são apresentadas no Quadro 12.1.

Quadro 12.1 **Palavras condicionais de loopings**

BEGIN-UNTIL
BEGIN-WHILE-REPEAT
DO-LOOP

As palavras (de desvio) de execução são apresentadas no Quadro 12.2.

Quadro 12.2 **Palavras condicionais de execução**

CASE:-THEN
IF-ELSE-THEN
IF-THEN

12.1 Fazendo comparações

Podemos comparar dois valores numéricos no sistema GraFORTH, conforme se vê no Quadro 12.3.

Quadro 12.3 **Palavras de comparação**

O FORMATO DAS PALAVRAS	COMPARAÇÃO EFETUADA
num1 num2 <>	num1 <> num2
num1 num2 =	num1 = num2
num1 num2 >	num1 > num2
num1 num2 <	num1 < num2
num1 num2 >=	num1 >= num2
num1 num2 <=	num1 <= num2

As palavras listadas no Quadro 12.3 instruem o sistema GraFORTH a retirar os dois últimos valores da pilha de dados e devolver o valor-verdade (1) ou valor-falso (0) de acordo com o resultado da comparação efetuada.

Analise os seguintes exemplos:

```
Ready  ( exemplos de comparação )
Ready  7  <>  5 .
1
Ready  10 BASE > .
0
Ready  0 VARIABLE VARIAVEL
Ready  VARIAVEL -21 >= .
0
Ready  BASE -> VARIAVEL
Ready  7 -> BASE
Ready  BASE VARIAVEL <= .
0
```

Essas palavras de comparação têm a sua maior aplicação na estrutura condicional de execução e laço.

Para estender as possibilidades de comparação, o sistema GraFORTH dispõe das palavras AND e OR que possuem os seguintes formatos:

```
num1  num2  AND
num1  num2  OR
```

Apresentamos os resultados possíveis das palavras AND e OR no Quadro 12.4.

Quadro 12.4 **Resultados das palavras AND e OR**

num1 \ num2	0	1
0	AND(0) OR(0)	AND(0) OR(1)
1	AND(0) OR(1)	AND(1) OR(1)

Vamos demonstrar então o uso das palavras AND e OR:

```
Ready  ABORT
Ready  7 DUP 1 > 2 = OR .
1
Ready  STACK
```

```
Ready  77 -21 OVER
[ 77 ]
[ -21 ]
[ 77 ]
Ready  (
[ 77 ]
[ 1 ]
Ready  SWAP 88
[ 1 ]
[ 77 ]
[ 88 ]
Ready  (
[ 1 ]
[ 1 ]
Ready  AND
[ 1 ]
Ready  ( portanto: -21(77(88 )
```

12.2 O laço DO-LOOP

As palavras DO e LOOP do sistema GraFORTH possibilitam executar um determinado bloco de palavras (num1 — num2) vezes. O formato do looping DO-LOOP é:

```
num1  num2  DO
      [ bloco-palavras ]
LOOP
```

onde o num1 é o valor do laço, o num2 é o valor inicial e o bloco-palavras é o bloco de palavras a ser executado (num1 — num2) vezes pelo sistema GraFORTH. O bloco-palavras pode estar presente ou não no formato, pois ele é opcional. Poderíamos entrar o seguinte exemplo sem problemas:

```
Ready  1000  0  DO  LOOP
```

Podemos representar o laço DO-LOOP graficamente pela figura 12.1.

Se você já conhece bem o laço FOR-NEXT do Applesoft BASIC, você pode notar que a estrutura do laço DO-LOOP do sistema GraFORTH é similar à estrutura do laço FOR-NEXT do Applesoft BASIC. Desejando conhecer melhor as instruções FOR-NEXT, você pode consultar o livro *Guia do Programador — Applesoft/Integer BASIC* da Hemus Editora Ltda.

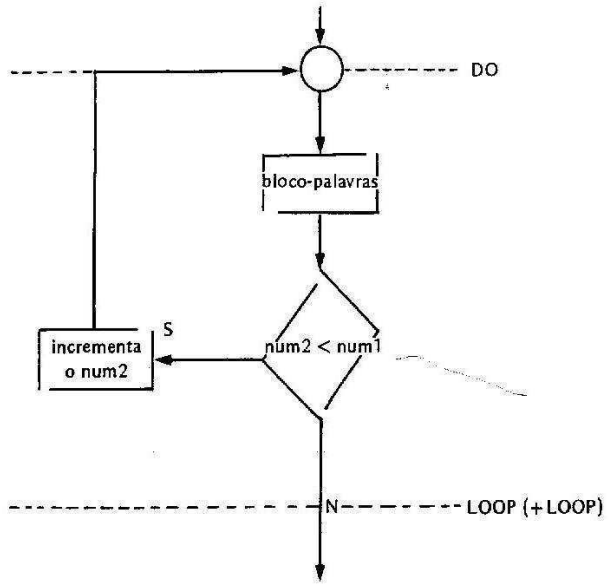


Figura 12.1 — A estrutura do laço DO-LOOP.

As palavras DO e LOOP devem ser entradas numa mesma linha ou dentro de uma definição de palavra.

O sistema GraFORTH dispõe de uma palavra (I) no dicionário que nos fornece o valor corrente da variação do laço. Comparando a estrutura do laço DO-LOOP com a do looping FOR-NEXT e usando os mesmos nomes no seu formato, teríamos:

```
FOR  I=num2 TO (num1 — 1):
     [ bloco-palavras: ]
NEXT
```

A palavra I do sistema GraFORTH então serve para fornecer o valor numérico corrente do laço, empilhando-o na pilha de dados. Vamos demonstrar então o uso do laço DO-LOOP:

```
Ready  ABORT
Ready  : IT ;
Ready  : LACO
         DO
            PRINT " NUMERO " SPCE I . CR
         LOOP
         PRINT " FIM " CR ;
```

```
Ready  7 1  LACO
NUMERO 1
NUMERO 2
NUMERO 3
NUMERO 4
NUMERO 5
NUMERO 6
FIM

Ready  FORGET IT
```

Analisemos o exemplo dado: o incremento do laço é de uma unidade. A palavra LOOP serve para testar o fim do laço, o qual acontece quando o valor do laço for maior ou igual ao num1.

Podemos ter um laço com o incremento maior ou menor do que 1. Basta usar a palavra +LOOP em vez de LOOP. A palavra +LOOP aguardará, na pilha de dados, o valor do incremento do laço. Entre, por exemplo:

```
Ready  ABORT
Ready  : IT ;
Ready  7 VARIABLE INCREMENTO
Ready  : LACO
         DO
            PRINT " NUMERO " SPCE I . CR
         INCREMENTO +LOOP
         PRINT " FIM " CR ;
Ready  18 1  LACO
NUMERO 1
NUMERO 8
NUMERO 15
FIM

Ready  -10 -> INCREMENTO
Ready  -40 -20  LOOP
NUMERO -20
NUMERO -30
FIM

Ready  FORGET IT
```

Podemos executar um laço sem bloco de palavras no seu formato. Esse laço simplesmente fica incrementando o seu valor até finalizar o laço DO-LOOP:

```
Ready  ( LACO PARA PAUSAS )
Ready  : IT ;
```



```
Ready  : PAUSA
           10500 0 DO LOOP ;
Ready  PRINT " ESTA MENSAGEM " SPCE
       PAUSA PRINT " SERA EXIBIDA" CR
       PAUSA PRINT " COM PAUSAS " CR
       PAUSA PAUSA
ESTA MENSAGEM SERA EXIBIDA
COM PAUSAS

Ready  FORGET IT
```

O sistema GraFORTH permite aninhar até três laços, um dentro do outro, com possibilidades de recuperar os valores dos laços.

Se aninharmos dois laços DO-LOOP, poderemos recuperar o valor do laço interno através da palavra I, da mesma maneira como a palavra J recupera o valor do laço externo. No formato da estrutura FOR-NEXT do Applesoft BASIC, teríamos:

```
FOR J=num2 TO (num1 – 1):
  [ bloco-palavras ]
  FOR I=num4 TO (num3 – 1):
    [ bloco-palavras: ]
  NEXT:
  [ bloco-palavras ]
NEXT
```

Vamos entrar o seguinte programa, demonstrando o aninhamento de dois laços como também o uso das palavras I e J para recuperar os valores dos laços:

```
A    ( AUTONUM )
10   ( TESTE DO ANINHAMENTO DE 2 LACOS )
20   FORGET IT
30   : IT ;
40   10 VARIABLE INCREMENTO
50   : LACODUPLO
60     20  0   DO
70         PRINT " LACO EXTERNO " SPCE
80         I . CR 6 HTAB
90         PRINT " LACOS INTERNOS " SPCE
100        2  0  DO
110            SPCE I . SPCE
120        LOOP
130        CR 2 HTAB
140        PRINT " EXECUTOU UM LACO I "
150        CR
```

```
160     INCREMENTO +LOOP
170     PRINT " EXECUTOU O LACO J " CR ;
180   CLOSE RUN
```

Saindo do editor de textos, podemos compilar este programa a partir do arquivo de trabalho (MEMRD) ou de um arquivo previamente salvo no disco (READ). Ao compilar (executar) esse programa, o sistema GraFORTH exibirá então as seguintes mensagens:

```
LACO EXTERNO 0
     LACOS INTERNOS 0 1
  EXECUTOU UM LACO I
LACO EXTERNO 10
     LACOS INTERNOS 0 1
  EXECUTOU UM LACO I
EXECUTOU O LACO J
```

Analogamente podemos aninhar três laços, recuperando os valores dos laços através das palavras I, J e K, sendo que o laço mais interno se associa com a palavra I.

12.3 A pilha de retorno

O laço DO-LOOP utiliza a pilha de retorno do sistema GraFORTH, a qual tem uma estrutura similar à da pilha de dados. A pilha de retorno pode armazenar até 128 números tal como a pilha de dados, embora você raramente precisará armazenar tantos valores.

A pilha de retorno no sistema GraFORTH serve para armazenar os valores dos laços DO-LOOP. Quando a palavra DO é executada, o sistema GraFORTH retira os dois últimos valores (num1 e num2) do topo da pilha de dados e coloca-os na pilha de retorno de forma que o valor do laço DO-LOOP (num2) fique no topo da pilha de retorno e o valor final do laço (num1) fique logo na penúltima posição da pilha de retorno. A palavra LOOP incrementa uma unidade ao valor do laço na pilha de retorno e compara-o ao valor final do laço armazenado na pilha de retorno. A palavra I copia o valor do laço da pilha de retorno, empilhando-o na pilha de dados. E quando o laço é completado, o sistema GraFORTH retira então esses dois valores da pilha de retorno.

O sistema GraFORTH possui as palavras PUSH, PULL e POP para permitir a manipulação dos dados da pilha de retorno. Essas palavras serão bastante utilizadas no decorrer das nossas exposições sobre o sistema GraFORTH. Vamos então descrevê-las no Quadro 12.5:

Quadro 12.5 Palavras básicas de Manipulação de pilha de retorno

PALAVRAS	DESCRIÇÃO
POP	Remove o número do topo da pilha de retorno.
PULL	Retira o número do topo da pilha de retorno, armazenando-o no topo da pilha de dados.
PUSH	Retira o número do topo da pilha de dados, armazenando-o no topo da pilha de retorno.

Vamos observar os efeitos dessas palavras na pilha de dados, entrando a seguinte rotina:

```
Ready  ABORT
Ready  STACK  12  1  64
[ 12 ]
[ 1 ]
[ 64 ]
Ready  PUSH
[ 12 ]
[ 1 ]
Ready  SWAP
[ 1 ]
[ 12 ]
Ready  PULL
[ 1 ]
[ 12 ]
[ 64 ]
```

Vejamos uma aplicação dessas palavras, executando o seguinte programa-teste:

```
A     ( AUTONUM )
10    ( testando as palavras PULL e PUSH )
20    FORGET IT
30    : IT ;
40    : LACO
50      OVER OVER
60      PUSH PUSH DO
70          PRINT " TESTE DAS PALAVRAS "
80          SPCE PRINT " PUSH E PULL " CR
90          PULL PULL SWAP DO
```

```
100              J . SPCE I . CR
110          LOOP
120      LOOP
130      PRINT " FIM DO TESTE " ;
140   : PRINCIPAL  7  0  LACO ;
CLOSE RUN
```

12.4 O laço BEGIN-UNTIL

O GraFORTH permite desenvolver a estrutura de programação BEGIN-UNTIL dentro do seu programa. A forma desta estrutura é:

```
BEGIN
     [ bloco-palavras ]
condição
UNTIL
```

onde o bloco-palavras é o bloco de palavras a ser executado dentro do laço e a condição é uma expressão lógica qualquer do GraFORTH.

A palavra BEGIN indica o início do laço BEGIN-UNTIL. O bloco de palavras dentro do laço é executado até que o valor lógico da condição seja diferente de zero.

Podemos representar, no diagrama de blocos, a estrutura do laço BEGIN-UNTIL do GraFORTH. Analise então a figura 12.2.

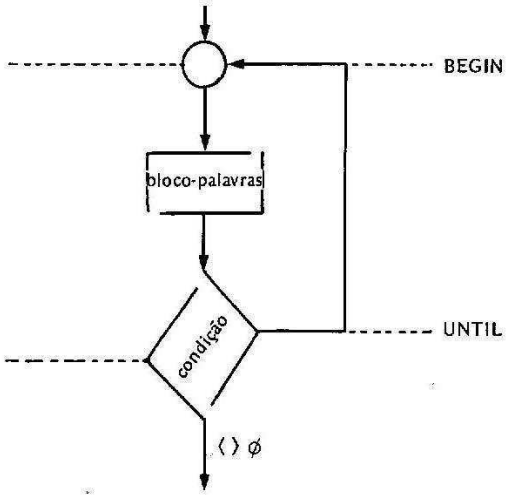


Figura 12.2 – A estrutura do laço BEGIN-UNTIL.

Teste o seguinte programa-exemplo, para analisar os efeitos das palavras BEGIN-UNTIL:

```
:  LACO.BU
   PRINT " CONTAGEM REGRESSIVA "
   ABS
   BEGIN
      DUP CR PRINT " NUMERO "
      SPCE .
      1 -
   DUP -1 =
   UNTIL
   CR PRINT " FIM " CR ;
```

Uma vez compilado o programa acima, execute a palavra LACO .BU já incluída no dicionário e obteremos o seguinte resultado na tela:

```
Ready  ABORT
Ready  STACK  77
[ 77 ]
Ready  10  LACO.BU
CONTAGEM REGRESSIVA
NUMERO  10
NUMERO  9
NUMERO  8
NUMERO  7
NUMERO  6
NUMERO  5
NUMERO  4
NUMERO  3
NUMERO  2
NUMERO  1
NUMERO  0
FIM
[ 77 ]
[ -1 ]
Ready  DROP
[ 77 ]
```

As palavras BEGIN e UNTIL devem ser entradas dentro de uma definição de palavra ou numa mesma linha.

12.5 O laço BEGIN-WHILE-REPEAT

A estrutura do laço BEGIN-WHILE-REPEAT do GraFORTH tem o seguinte formato:

```
BEGIN
     [ bloco-palavras-1 ]
condição
WHILE
     [ bloco-palavras-2 ]
REPEAT
```

onde a condição é uma expressão lógica qualquer do GraFORTH, o bloco-palavras-1 é geralmente o bloco de palavras a ser executado dentro do laço e o bloco-palavras-2 é o bloco de palavras que geralmente visa a alterar e controlar o valor lógico da condição.

As palavras BEGIN, WHILE e REPEAT devem ser entradas dentro de uma definição de palavra ou numa mesma linha.

A palavra BEGIN indica o início de um laço BEGIN-WHILE-REPEAT. O bloco-palavras-1 e a expressão lógica de condição são então executados; a palavra WHILE retirará um número da pilha de dados e compara-lo-á com o valor zero. Se esse número for diferente de zero, o GraFORTH executará o bloco-palavras-2 e retornará a execução seqüencial desse laço a partir da palavra BEGIN. Se esse número for nulo, o sistema GraFORTH termina a execução do laço BEGIN-WHILE-REPEAT, passando a executar a próxima palavra do programa ou aguardará a sua próxima entrada, exibindo o prompt Ready. Podemos representar de modo gráfico a estrutura do laço BEGIN-WHILE-REPEAT pela figura 12.3.

Modificando o programa "LACO.BU", adaptando-o à estrutura do laço BEGIN-WHILE-REPEAT, temos o seguinte:

```
: LACO.BWR
  PRINT " CONTAGEM REGRESSIVA "
  ABS
  BEGIN
    DUP CR PRINT " NUMERO "
    SPCE .
  DUP 0 >
  WHILE
    1 -
  REPEAT
  CR PRINT " FIM " CR ;
```

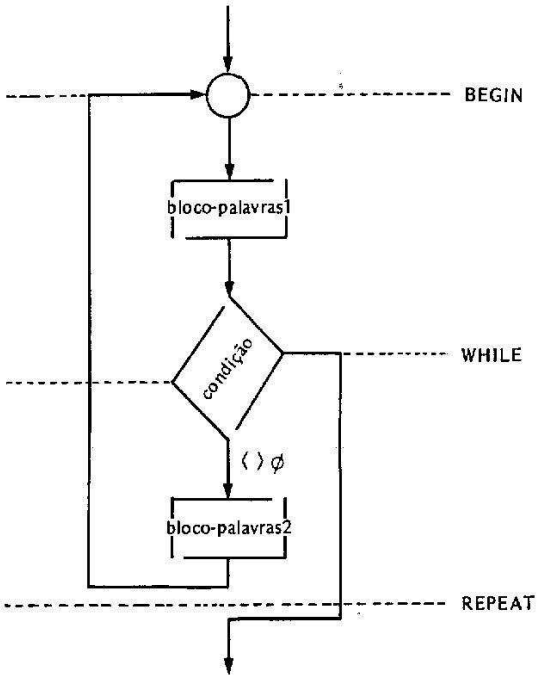



Figura 12.3 – A estrutura do laço BEGIN-WHILE-REPEAT.

Execute a palavra LACO.BWR já incluída no dicionário e compare com a execução da palavra LACO.BU listada no item 10.4 deste capítulo:

```
Ready  ABORT
Ready  STACK  77
[ 77 ]
Ready  10  LACO.BWR
CONTAGEM REGRESSIVA
NUMERO 10
NUMERO 9
NUMERO 8
NUMERO 7
NUMERO 6
NUMERO 5
NUMERO 4
```

```
NUMERO 3
NUMERO 2
NUMERO 1
NUMERO 0
FIM
[ 77 ]
```

12.6 A estrutura de decisão IF-THEN

A estrutura de decisão mais simples é a estrutura condicional IF-THEN cujo formato é:

```
condição IF
      [ bloco-palavras ]
THEN
```

As palavras IF e THEN devem ser entradas numa única linha ou dentro de uma definição de uma nova palavra.

A palavra IF retira um valor lógico colocado no topo da pilha de dados pela expressão booleana condição e compara-o com o número zero. Se esse número for diferente de zero, o bloco-palavras será então executado. Se esse número for nulo, o bloco-palavras será ignorado e o GraFORTH então executará a próxima linha do seu programa ou aguardará suas instruções, exibindo o prompt Ready. Na figura 12.4 temos a representação da estrutura de decisão IF-THEN do GraFORTH num diagrama de blocos.

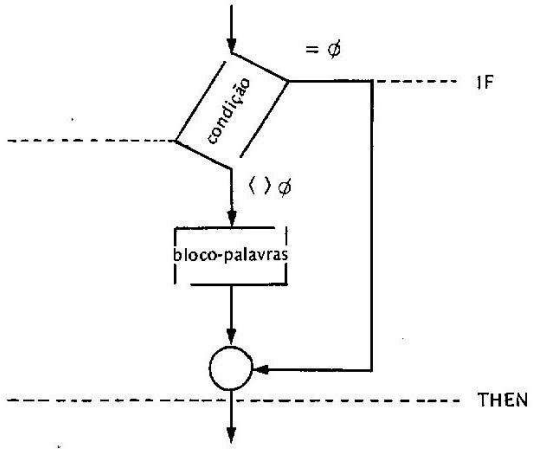


Figura 12.4 — A estrutura de decisão IF-THEN.

No decorrer das explanações do livro, vamos demonstrar as mais variadas aplicações para as palavras IF-THEN. Neste item, tente apenas testar e compreender a estrutura de decisão IF-THEN, entrando o seguinte programa:

```
FORGET IT
: IT ;
: TESTE1
  1 = IF
    PRINT " POSITIVO " CR
  THEN ;
: TESTE2
  0 = IF
    PRINT " NULO " CR
  THEN ;
: TESTE3
  -1 = IF
    PRINT " NEGATIVO " CR
  THEN ;
: TESTE
  DUP PRINT " O NUMERO " SPCE . SPCE
  PRINT " E' " SPCE
  SGN
  DUP DUP DUP
  TESTE1 TESTE2 TESTE3
  PRINT " FIM " CR ;
: PRINCIPAL
  HOME
  PRINT " FORMATO: NUM TESTE " CR
  -10 TESTE
  867 TESTE
  77 100 / TESTE ;
CLOSE RUN
```

12.7 A estrutura de decisão IF-ELSE-THEN

A estrutura condicional IF-ELSE-THEN é uma variação da estrutura de decisão IF-THEN e tem o formato:

```
condição IF
      [ bloco-palavras-1 ]
ELSE
      [ bloco-palavras-2 ]
THEN
```

As palavras IF, ELSE e THEN devem ser entradas ou numa única linha ou dentro de uma definição de uma nova palavra.

A palavra IF retira um valor lógico colocado no topo da pilha de dados pela expressão booleana condição e compara-o com o número zero. Se esse número for diferente de zero, o bloco-palavras-1 será executado. Se esse número for nulo, o bloco-palavras-2 será executado. Na figura 12.5 temos a representação da estrutura de decisão IF-ELSE-THEN do GraFORTH num diagrama de blocos.

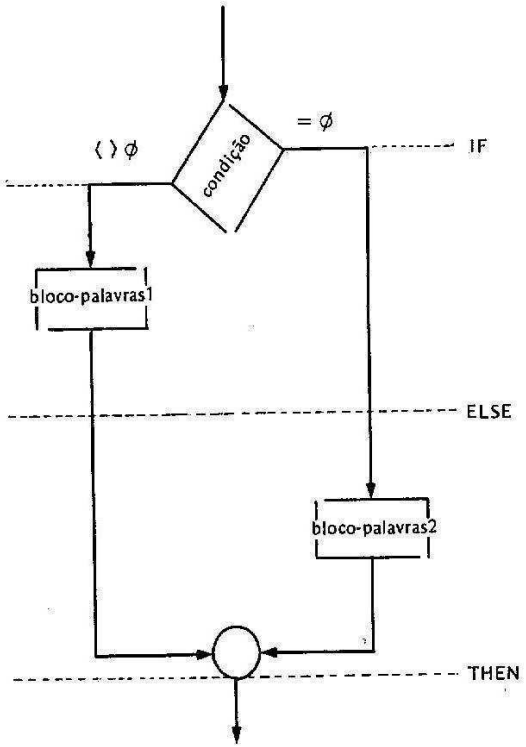


Figura 12.5 — A estrutura de decisão IF-ELSE-THEN.

No decorrer deste livro vamos demonstrar as mais variadas aplicações para as palavras IF-ELSE-THEN. Tente, neste item, apenas testar e compreender a estrutura de execução condicional IF-ELSE-THEN, entrando o seguinte programa:

```
FORGET IT
: IT ;
: TESTE
  DUP PRINT " O NUMERO " SPCE . SPCE
  PRINT " E' " SPCE
  SGN DUP
  1 = IF
    PRINT " POSITIVO " CR
  ELSE
    -1 = IF
      PRINT " NEGATIVO " CR
    ELSE
      PRINT " NULO " CR
    THEN
  THEN
  PRINT " FIM " CR ;
: PRINCIPAL
  HOME
  PRINT " FORMATO: num TESTE " CR
  -10 TESTE
  867 TESTE
  77 100 / TESTE ;
CLOSE RUN
```

12.8 A estrutura de decisão CASE:-THEN

A estrutura de execução condicional CASE:-THEN tem o seguinte formato:

```
condição CASE:
      palavra-0
      palavra-1
      palavra-2
         .
         .
         .
      palavra-n
THEN
```

A estrutura de decisão CASE:-THEN permite executar uma determinada palavra dentro de uma faixa de números de palavras possíveis.

As palavras CASE: e THEN devem ser entradas dentro de uma definição de palavra ou numa mesma linha.

A palavra CASE: retira um número colocado na pilha de dados pela expressão condição e usa-o para selecionar a execução de uma única palavra dentro da lista de palavras definidas, de forma que o número 0 seleciona a palavra-0, o número 1 seleciona a palavra-1, e assim por diante. A palavra THEN é necessária para indicar o fim da estrutura de execução condicional CASE:-THEN. Na figura 12.6, temos a representação da estrutura de decisão CASE:-THEN num diagrama de blocos.

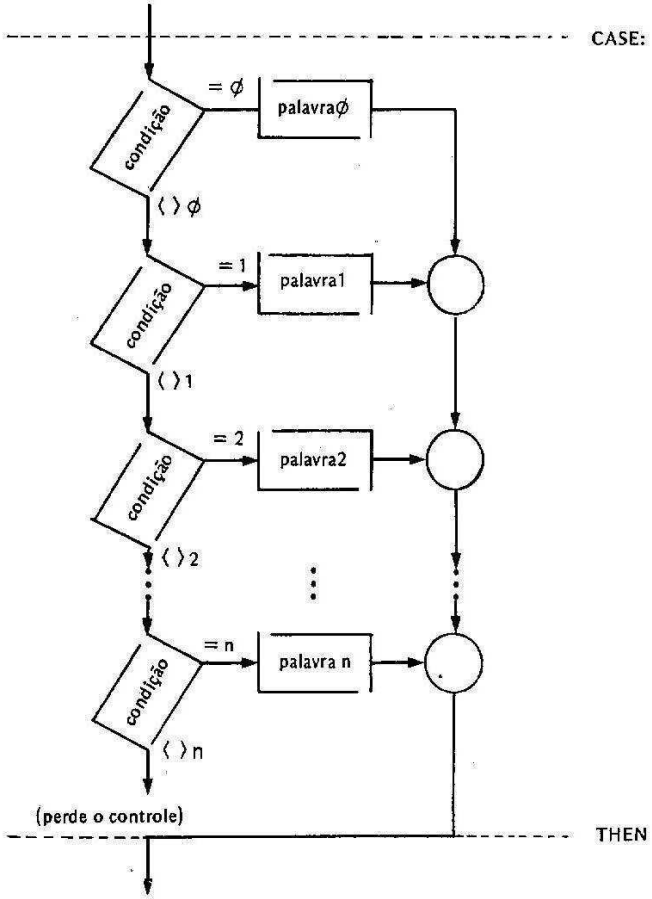


Figura 12.6 — A estrutura de decisão CASE:-THEN.

Demonstraremos a aplicação e a variação desta estrutura de decisão, apresentando o seguinte programa, similar àquele apresentado no item 12.7 deste capítulo.

```
FORGET IT
: IT ;
: NEGATIVO
  PRINT " NEGATIVO " CR ;
: NULO
  PRINT " NULO " CR ;
: POSITIVO
  PRINT " POSITIVO " CR ;
: TESTE
  DUP PRINT " O NUMERO " SPCE . SPCE
  PRINT " E' " SPCE
  SGN
  1 + CASE:
    NEGATIVO
    NULO
    POSITIVO
  THEN
  PRINT " FIM " CR ;
: PRINCIPAL
  HOME
  PRINT " FORMATO: num TESTE " CR
  -10  TESTE
  867  TESTE
  77 100 / TESTE ;
CLOSE RUN
```

Comparando o programa TESTE deste item com os dos itens anteriores, notamos que este programa-exemplo aqui apresentado possui uma estrutura de programação mais nítida e melhor distribuída que as estruturas das versões anteriores deste programa.

12.9 Uma consideração sobre o sistema GraFORTH

O sistema GraFORTH pode executar as mais variadas aplicações com uma velocidade incrível. Essa rapidez de compilação e de execução é melhor evidenciada na manipulação dos gráficos tridimensionais. Como para tudo há um preço, o sistema GraFORTH é limitado a: operar valores em uma faixa restrita de números inteiros; verificar os erros de sintaxe com menos recursos e freqüências possíveis, pois a verificação de erros implica execução de mais rotinas; desenhar e animar seus gráficos com uma quantidade de pontos menor do que

a quantidade real permitida pelo hardware (ver o Capítulo 14). Contudo, mesmo com essas restrições, o sistema GraFORTH permite recursos muito flexíveis e poderosos, além de sua incrível velocidade.

12.10 A verificação de erros

O sistema GraFORTH possui poucas rotinas de verificação de erros de compilação e de execução para verificar se o usuário tentou esquecer uma palavra que existe ou não no dicionário, tais como: se o estado da pilha de dados é normal ou não, e se ocorreu o fenômeno de overflow ou underflow; ou se os elementos de um bloco de palavras são compatíveis (rever o caso da VARIABLE no item 8.3 do Capítulo 8) etc. Algumas rotinas de verificação de erros são suficientes. Por exemplo, entre:

```
Ready : IMP1 CR PRINT ;
Ready : IMP2 SPCE PRINT ;
Ready IMP1 " TESTE1 " IMP2 " TESTE2 "
```

A princípio, executar a rotina acima parece uma boa idéia. Contudo, nunca a experimente novamente, pois quando você compilar o bloco de palavras : IMP1 CR PRINT ;, o sistema GraFORTH montará no dicionário de palavras a palavra IMP1 com erros (porém, passa pela compilação sem nenhuma mensagem de erros), e para você entender o erro cometido, você deverá entender melhor a estrutura do dicionário e as palavras do sistema GraFORTH.

Vamos analisar um outro erro de compilação do sistema GraFORTH:

```
Ready : + OVER OVER . SPCE . SPCE + . ;
+ Not Unique (Return)
```

Como a palavra + já foi empilhada no dicionário — aliás, a palavra + é uma palavra primitiva do dicionário —, temos depois de gerar esta mensagem de erro, duas palavras iguais no dicionário. Se executarmos a palavra + agora, você verá um looping de exibição de números interminável em execução. Pressione as teclas ⟨CTRL⟩⟨RESET⟩ e esqueça a segunda palavra +, entrando apenas:

```
Ready FORGET +
```


13

APROFUNDANDO-SE NA PROGRAMAÇÃO COM O GRAFORTH

13.0 Manipulando a memória

As palavras do dicionário do sistema GraFORTH, listadas no Quadro 13.1, permitem examinar e manipular "qualquer" uma das posições de memória do sistema Apple II. Você deve saber que cada uma dessas posições de memória pode armazenar somente um "byte de 8 bits", representando os valores 0 a 255. De uma forma mais limitada, as palavras que tratam das tabelas, das variáveis strings e das variáveis numéricas também permitem manipular uma faixa da memória.

Quadro 13.1 **Palavras básicas de manipulação de memória**

FORMATO DAS PALAVRAS	DESCRIÇÃO
num PEEK	Lê um valor de um byte da posição num.
num PEEKW	Lê um valor de dois bytes das posições num e (num+1).
num1 num2 POKE	Armazena o valor num1 de um byte na posição num2.
num1 num2 POKEW	Armazena o valor num1 de dois bytes nas posições num2 e (num2+1).
num1 num2 num3 MOVMEM	Move num3 bytes da posição num1 para a posição num2.

Neste item, vamos explicar apenas o formato das palavras do Quadro 13.1. No decorrer do livro, descreveremos todas as aplicações possíveis para essas palavras.

13.0.0 Endereçamento de memória do sistema Apple II

Como se sabe, o sistema Apple II pode endereçar até 65.536 posições de memória, numeradas de 0 a 65.535, e no sistema GraFORTH você pode manipular (ler/gravar) diretamente a maioria dessas posições de memória através das palavras mostradas no Quadro 13.1.

Para endereçar as 65.536 posições de memória do Apple II, o sistema GraFORTH usará os números -32.768 até 32.767 para endereçar as posições de memória 0 a 65.535, tal como os interpretadores Applesoft e Integer BASIC trabalham (lembre-se da instrução CALL-151). Elaboramos o Quadro 13.2 para lhe mostrar a correspondência entre os números do sistema GraFORTH e o endereço do seu Apple II:

Quadro 13.2 **Modo de endereçamento do GraFORTH**

Endereçamento decimal positivo	Endereçamento decimal do GraFORTH
0	0
1	1
2	2
.	.
.	.
.	.
32765	32765
32766	32766
32767	32767
32768	−32768
32769	−32767
32770	−32766
.	.
.	.
.	.
65533	−3
65534	−2
65535	−1

O modo de endereçamento permitido pelo sistema GraFORTH deve-se à sua limitação de operar com a faixa de números inteiros de -32768 a 32767. Contudo, você pode referenciar os endereços da

memória com valores decimais positivos, pois o sistema GraFORTH converterá esses valores entrados em valores na faixa permitida pelo GraFORTH. Por exemplo, entre:

```
Ready ABORT
Ready STACK
Ready 65535
[ -1 ]
Ready 32768
[ -32768 ]
```

No Apêndice, fornecemos um mapa de memória do seu Apple II com o sistema GraFORTH instalado.

13.0.1 A palavra POKE

Conforme vimos no Quadro 13.1, o formato da palavra POKE é:

```
num1 num2 POKE
```

onde o num1 é o valor (de 0 a 255) a ser armazenado na posição num2 do Apple II. Por exemplo, entre:

```
Ready 255 2816 POKE
```

A palavra POKE é muito utilizada para armazenar um determinado valor representando um caractere ou número numa posição específica de memória. Podemos também montar uma sub-rotina em linguagem de máquina numa determinada área de memória do sistema Apple II, para posterior execução pela palavra CALL.

13.0.2 A palavra PEEK

A palavra PEEK possibilita examinar uma posição da memória do seu Apple II. Seu formato é:

```
num PEEK
```

onde o número num indica o endereço contendo o valor a ser lido e colocado no topo da pilha.

Podemos ler um valor (de 0 a 255) de uma posição de memória, compará-lo com um determinado valor e modificar o conteúdo dessa posição de memória:

```
FORGET IT
: IT ;
: TESTE.POKE
160   2816  POKE
250   2817  POKE ;
: TESTE.PEEK
  DO  LOOP
  2816  PEEK
  2817  PEEK
  OVER <> IF
      PRINT " CODIGO " SPCE DUP .
      20 HTAB  PRINT " CARACTER "
      SPCE  PUTC  CR
      2816 PEEK  1  +  2816  POKE
      TESTE.PEEK
  THEN ;
: PRINCIPAL
  INVERSE
  PRINT " OS CARACTERES E " SPCE
  PRINT " VALORES ASCII "
  NORMAL  CR  CR
  TESTE.POKE
  1000  0  TESTE.PEEK ;
CLOSE RUN
```

Com a palavra PEEK, podemos controlar determinados recursos do GraFORTH, por exemplo:

```
FORGET IT
: IT ;
: VERIF.CURSOR.HORIZONTAL
  36  PEEK  CR ;
: PRINCIPAL
  PRINT " VERIFICANDO A ATUAL POSICAO "
  SPCE PRINT " HORIZONTAL "
  INVERSE PRINT " ) " NORMAL SPCE
  VERIF.CURSOR.HORIZONTAL  .
  DUP . 7 + 36 POKE  ( PULANDO 7 COLUNAS )
  PRINT " FIM " ;
  CLOSE RUN
```

13.0.3 A palavra POKEW

A palavra POKEW possibilita armazenar um número inteiro qualquer (de -32768 a 32767) em duas posições de memória seguidas.

Seu formato é:

```
num1  num2  POKEW
```

onde o num1 é o valor a ser armazenado nos endereços num2 e (num2 + 1). Para entender melhor a execução da palavra POKEW, simulamos sua estrutura lógica através da palavra POKEW2:

```
: POKEW2
  ( simulando a palavra POKEW )
  SWAP  DUP  DUP
  SGN  -1   IF
       ( dividindo o valor entrado )
       ( em dois bytes )
       256  /  255  +
       SWAP
       CHS  256 MOD CHS  256 +
       DUP 256  =  IF
            DROP  0
            SWAP  1  +  SWAP
       THEN
  ELSE
       256  /
       SWAP
       256  MOD
  THEN
  4   PICK  POKE
       ( armazenando o byte menos )
       ( significativo )
  3   PICK  1  +  POKE
       ( armazenando o byte mais )
       ( significativo )
  DROP ;
```

13.0.4 A palavra PEEKW

Podemos ler um valor (de -32768 a 32767) de qualquer das duas posições de memória. O formato da palavra PEEKW é o seguinte:

```
num  PEEKW
```

onde o num é o endereço que contém o valor do byte menos significativo e o (num + 1) é o endereço que contém o valor do byte mais significativo de um número. Cada um dos números em GraFORTH é

definido em função de dois bytes, de forma que esses dois bytes possam representar todos os valores através da seguinte expressão:

256 * (byte-mais-significativo) + (byte-menos-significativo)

Vamos observar melhor a estrutura lógica da palavra PEEKW, analisando a palavra PEEKW2, a qual visa simular as funções da palavra PEEKW:

```
: PEEKW2
  ( simulando a palavra PEEKW )
  DUP
  PEEK
  ( lendo o byte menos significativo )
  SWAP
  1 + PEEK
  ( lendo o byte mais significativo )
  255 * +
  ( empilhando, no topo da pilha, )
  ( o valor do número lido ) ;
```

13.0.5 A palavra MOVMEM

A palavra MOVMEM simplesmente move (copia) um bloco de num3 bytes para a posição de memória num2 a partir da posição num1. Assim, o formato para a palavra MOVMEM é:

num1 num2 num3 MOVMEM

A estrutura lógica e funcional da palavra MOVMEM pode ser simulada pela palavra MOVMEM2:

```
: MOVMEM2
  ( simulando a palavra MOVMEM2 )
  0 DO
       OVER  OVER  SWAP
       PEEK
       SWAP  POKE
       1  +  SWAP
       1  +  SWAP
  LOOP
  DROP DROP ;
```


13.1 As variáveis strings

As variáveis strings, em GraFORTH, são as palavras do dicionário que servem para armazenar um conjunto de caracteres alfanuméricos ou um conjunto de valores (de 0 a 255).

As variáveis strings são geralmente usadas quando se deseja: entrar dados pelo teclado durante a execução de um programa, manipular determinados textos, manipular valores específicos na faixa de memória delimitada pela variável string, ou montar uma determinada instrução durante a execução de um programa a fim de que esta palavra possa ser compilada e executada.

13.1.1 Definindo as variáveis strings

No sistema GraFORTH, a palavra STRING é usada para definir as variáveis strings e possui o seguinte formato:

comprimento STRING palavra-string

onde a palavra-string é o nome da palavra a ser definida como uma variável string e o comprimento é o número máximo de caracteres (bytes) que a variável string aqui definida pode armazenar.

Execute a palavra LIST antes e depois de entrar a seguinte instrução:

```
Ready 100 STRING TESTE
```

Como as variáveis strings são palavras em GraFORTH, elas podem também ser esquecidas. Experimente:

```
Ready FORGET TESTE
```

13.1.2 Acessando-se às variáveis strings

No sistema GraFORTH, pode-se acessar à variável string a partir do num-ésimo caractere, ou seja, podemos ler/armazenar caracteres alfanuméricos numa variável string a partir do seu num-ésimo byte disponível. A variável string, portanto, deve ser referenciada nesse formato:

posição palavra-string bloco-palavras

onde posição indica o num-ésimo byte disponível na palavra-string e o bloco-palavras é o bloco de palavras que manipula os dados nessa

variável string. A expressão "posição palavra-string" instrui o GraFORTH a colocar na pilha o endereço da num-ésima posição de memória do seu Apple II correspondente ao num-ésimo byte da palavra-string. Observe o seguinte programa:

```
Ready  ABORT
Ready  STACK
Ready  100   STRING  TESTE
       ( definindo a variável string )
       ( TESTE com 100 caracteres )
Ready  0   TESTE
       ( acessando o primeiro caractere )
       ( da variável string TESTE )
[-31930 ]
Ready  ASSIGN " TEXTO ARMAZENADO "
      ( armazenando o texto na variável )
      ( string através do seu endereço )
Ready  0   TESTE
[-31930 ]
Ready  WRITELN
      ( exibindo o conteúdo da variável )
      ( string em códigos ASCII )
TEXTO ARMAZENADO
```

A instrução 0 TESTE faz com que o sistema GraFORTH coloque, no topo da pilha, o endereço do primeiro byte da string TESTE.

Como você pode observar, as palavras ASSIGN, READLN e WRITELN aguardam na pilha o endereço -31930 (do primeiro byte da variável string a operar), colocado na pilha de dados pela instrução 0 TESTE.

Observação: os endereços aqui fornecidos podem não ser os mesmos exibidos na execução desse exemplo no seu Apple II.

13.1.3 Usando as variáveis strings

Vamos analisar, neste item, as palavras primitivas do sistema GraFORTH que operam com os endereços das palavras-strings (variáveis strings) e os códigos ASCII. Essas palavras (ver o Quadro 13.3) visam facilitar a entrada e a saída de caracteres alfanuméricos, propondo uma melhor interação homem-máquina com esses recursos.

Quadro 13.3 Palavras básicas de manipulação de variáveis strings

PALAVRAS	DESCRIÇÃO
ASSIGN	Armazena uma seqüência de caracteres alfanuméricos numa variável string.
PEEK	Coloca no topo da pilha o valor em código ASCII de um caractere alfanumérico numa variável string.
POKE	Retira um valor do topo da pilha e armazena-o como um valor ASCII numa variável string.
READLN	Lê uma seqüência de caracteres alfanuméricos a partir do teclado, armazenando-a numa variável string.
WRITELN	Exibe todo o conteúdo ou uma parte de uma variável string.

13.1.3.0 A palavra ASSIGN

Você pode armazenar diretamente uma string (uma seqüência de caracteres alfanuméricos) numa palavra-string (variável string), usando a palavra ASSIGN do dicionário do sistema GraFORTH, no formato:

```
endereço-string  ASSIGN  " texto-string "
```

onde o endereço-string é uma posição da memória do seu sistema Apple II, em que você quer armazenar o texto-string (uma seqüência de caracteres alfanuméricos). Exemplificando o formato da palavra ASSIGN:

```
Ready  20   STRING  PALAVRA
Ready  0   PALAVRA
       ASSIGN " 1001 0 DO I . LOOP "
```

Neste exemplo, criamos uma palavra string com um comprimento de 20 bytes e armazenamos nela uma seqüência de caracteres alfanuméricos. Agora, se você conseguir se lembrar do uso da palavra MEMRD, podemos entrar:

```
Ready  0  PALAVRA  MEMRD
```

A palavra MEMRD compilará e executará o texto (instruções) que estiver armazenado no primeiro byte da variável string PALAVRA.

Esta é uma das melhores aplicações possíveis fornecidas pela variável string.

13.1.3.1 A palavra WRITELN

Para exibir, na tela do seu sistema Apple II, os caracteres alfanuméricos armazenados numa variável string a partir de uma determinada posição da palavra-string, usa-se a palavra WRITELN no formato:

endereço-string WRITELN

A palavra WRITELN aguarda, no topo da pilha, um endereço qualquer de memória, interpreta os valores consecutivos na faixa de memória após o endereço dado como sendo códigos ASCII e exibe os seus respectivos caracteres em valores ASCII na tela até encontrar o valor 141 (em código ASCII, equivale ao caractere ConTRoL-M). Ciente da lógica que orienta a execução da palavra WRITELN, podemos simulá-la, criando a palavra WRITELN2:

```
: WRITELN2
  ( simulando a palavra WRITELN )
  DUP PEEK
  DUP 141 <> IF
      PUTC
      1 +
      WRITELN2
  THEN ;
```

Vamos então executar a palavra WRITELN:

```
A     ( AUTONUM )
10    FORGET IT
20    : IT ;
30    20 STRING VARIAVEL$
40    0 VARIAVEL$ ASSIGN " LETRA x "
50    : TESTE
60      PRINT " PRESSIONE UMA TECLA "
70      SPCE GETC
        ( obtendo uma tecla )
80    6 VARIAVEL$ POKE
      ( armazenando a letra pressionada )
90    0 VARIAVEL$ WRITELN2 ;
100   CLOSE RUN
```


13.1.3.2 A palavra READLN

A palavra READLN armazena uma seqüência de caracteres alfanuméricos e símbolos especiais entrados pelo teclado durante a sua execução. Assemelha-se à instrução INPUT do Applesoft BASIC. Seu formato é:

endereço-string READLN

A palavra READLN aguarda no topo da pilha um endereço da memória do seu Apple II; exibe o cursor, lê uma linha de textos a partir do teclado, exibindo os caracteres entrados; e armazena os caracteres em códigos ASCII a partir da posição de memória dada. Por exemplo:

```
FORGET IT
: IT ;
40   STRING INPUT$
: TESTE.READLN
  PRINT " ENTRE O QUE VOCE QUISER " SPCE
  0 INPUT$ READLN ;
CLOSE RUN
```

Você deve tomar cuidado com as palavras READLN e ASSIGN, pois se o usuário do seu programa tentar entrar uma seqüência de caracteres maior do que o comprimento da palavra-string referenciada, o sistema GraFORTH perderá o total controle de execução. Exemplificando:

```
Ready 10 STRING TESTE$
Ready CR 0 STRING READLN
( entre os seguintes números na )
( execução da palavra READLN )

123456789012343455
Ready ( você perdeu o controle )
```

Podemos também simular a lógica de execução da palavra READLN, criando a palavra READLN2:

```
FORGET IT
: IT ;
: READLN2
  ( obtendo um caractere do teclado)
  GETC DUP PUTC ( exibindo a letra )
  DUP <> 141 IF     ( se <> ConTRoL-M )
```

```
   OVER POKE     ( armazena a letra )
   1 +           ( e leia a próxima )
   READLN2       ( tecla. )
THEN ;
```

13.1.3.3 A palavra GETNUM

O sistema GraFORTH permite converter uma seqüência de caracteres numéricos armazenados numa variável string em valores numéricos. O formato da palavra responsável por esse recurso é:

endereço-string GETNUM

A palavra GETNUM retira um valor do topo do dicionário, interpreta-o como um endereço de uma variável string, e converterá em valores numéricos todos os caracteres numéricos em códigos ASCII, armazenados na seqüência de bytes a partir do endereço-string.

Você pode também testar se uma seqüência de posições de memória do seu Apple II armazena ou não uma seqüência de caracteres numéricos em códigos ASCII, entrando a palavra VALID após a GETNUM.

A palavra VALID colocará no topo da pilha um valor nulo, se a palavra GETNUM não for capaz de converter em número nenhuma seqüência de caracteres referenciada. Caso contrário, empilhará um valor não-nulo. Por exemplo:

```
FORGET NUM$
FORGET IT
: IT ;
30 STRING NUM$
: TESTE.NUM
   PRINT " ENTRE UM NUMERO QUALQUER -> "
   SPCE 0 NUM$ READLN
   0 NUM$ GETNUM DUP
   VALID 0 = IF
      PRINT " ENTRE SOMENTE NUMEROS " CR
      TESTE.NUM
   THEN
   PRINT " O NUMERO ENTRADO -> " SPCE .
   CR PRINT " FIM DO TESTE.NUM " ;
```

13.2 A palavra PAD – A variável string do sistema GraFORTH

A palavra PAD é uma variável string do sistema GraFORTH com 124 caracteres de comprimento e muito usada para armazenar algum

dado a entrar pelo teclado ou para conter uma instrução a ser compilada.

O primeiro endereço da variável string PAD é 812, visto que a palavra PAD retorna com esse valor. Porém, a área de armazenamento da variável string PAD é diferente da área da definição da palavra PAD no dicionário. Usamos a palavra PAD para referenciar uma palavra do dicionário do sistema GraFORTH, que, quando executada, retorna com o valor da primeira posição de uma área livre de 124 bytes de memória para armazenar strings, chamada de variável PAD. Diferentemente das variáveis strings definidas pela palavra STRING, a variável string PAD não armazena as strings na própria área de memória da definição da palavra PAD, pois a palavra PAD e a variável string PAD são distintas. O formato da palavra PAD é: PAD

Você pode notar pelo formato da palavra PAD que ela não permite a indexação normal da posição dos seus bytes tal como ocorre em outras variáveis strings. Por exemplo:

```
Ready ABORT
Ready STACK
Ready PAD
[ 812 ]
Ready ASSIGN " 0123456789ABCDEF "
Ready PAD 13 +
[ 825 ]
Ready WRITELN
DEF
```

Poderíamos indexar os endereços da variável string PAD, criando a palavra PAD2:

```
Ready : PAD2
          PAD + ;
Ready 0 PAD2 ASSIGN " 0123456789ABCDEF "
Ready 14 PAD2 WRITELN
EF
```

A variável string PAD é considerada uma área temporária para armazenar strings, visto que essa área é também requerida pelo sistema GraFORTH para compilar as definições de novas palavras, servindo como uma área auxiliar do sistema (ver o Apêndice).

13.3 O arquivo de textos STRING WORDS

O arquivo de textos STRING WORDS do disquete do sistema GraFORTH contém palavras adicionais para manipulação das variá-

veis strings com recursos mais complexos. Para incluir essa biblioteca de novas palavras no dicionário do sistema, basta compilar esse arquivo:

```
Ready  READ " STRING WORDS "
```

Compilando do arquivo STRING WORDS, teremos as novas palavras mostradas no Quadro 13.4.

Quadro 13.4 **Palavras do arquivo STRING WORDS**

PALAVRAS	DESCRIÇÃO
COMPARE	Compara os conteúdos das duas variáveis strings especificadas.
CONCAT	Copia uma string de uma variável string no final da outra, concatenando-as.
END?	Verifica se a atual posição memória é o fim da variável string referenciada.
LENGTH	Retorna com o comprimento da variável string a partir do endereço dado.
LEFT$	Armazena numa outra uma seqüência de caracteres mais à esquerda de uma variável string.
MOVELN	Copia numa outra uma string de uma variável string.
RIGHT$	Armazena numa outra uma seqüência de caracteres mais à direita de uma variável string.

13.3.0 A palavra END?

A palavra END? da biblioteca STRING WORDS permite determinar se uma dada posição da variável string contém o caractere ConTRoL-M (o valor 141 do código ASCII) ou o valor 0 do código ASCII. Se um determinado endereço-string contiver os valores 0 ou 141, a palavra END? retorna com o valor verdade 1. Se não, devolve o valor falso 0. O formato da palavra END? é então:

```
endereço-string  END?
```


Demonstrando o uso da palavra END?:

```
FORGET IT
: IT ;
20  STRING TESTE
: TESTE.END?
  PRINT " ENTRE UMA PALAVRA "
  0 TESTE READLN
  21 0 DO
    I TESTE
    END? 0 = IF
         I TESTE PEEK
         PUTC
    THEN
  LOOP
  PRINT " FIM " ;
```

13.3.1 A palavra LENGTH

Formato: endereço-string LENGTH

Entrando a palavra LENGTH no formato acima, ela retorna com o valor do comprimento da variável string referenciada a partir do endereço-string dado. Experimente:

```
FORGET IT
: IT ;
20  STRING TESTE
: TESTE.LENGTH
  21 0 DO
     I TESTE LENGTH
     PRINT " COMPRIMENTO -> " SPCE .
     CR
  LOOP ;
```

13.3.2 A palavra LEFT$

Formato: endereço-string1 endereço-string2 LEFT$

A palavra LEFT$ copia uma seqüência de caracteres alfanuméricos mais à esquerda de uma variável string a partir do endereço-string1 para uma outra variável a partir do endereço-string2. A palavra LEFT$ da biblioteca STRING WORDS é semelhante à função LEFT$ do Applesoft BASIC. Entre:

```
Ready  40 STRING VAR$
```

```
Ready  0  VAR$ ASSIGN " MEU APPLE II "
       0  VAR$ PAD  10 LEFT$
       PAD WRITELN
MEU APPLE
```

13.3.3 A palavra RIGHT$

Formato: endereço-string1 endereço-string2 RIGHT$

A palavra RIGHT$ copia uma seqüência de caracteres alfanuméricos mais à direita de uma variável string a partir do endereço-string1 para uma outra variável a partir do endereço-string2. A palavra RIGHT$ da biblioteca STRING WORDS é similar à função RIGHT$ do Applesoft BASIC. Experimente:

```
Ready  12  STRING VAR$
Ready  0  VAR$  ASSIGN " MEU APPLE II "
       0  VAR$  PAD  8  RIGHT$
       PAD WRITELN
APPLE II
```

13.3.4 A palavra MOVELN

A palavra MOVELN da biblioteca STRING WORDS duplica uma string de uma posição de memória numa outra. O formato desta palavra é:

endereço-string1 endereço-string2 MOVELN

onde o endereço-string1 é a localização a partir da qual está contida uma string a ser copiada para uma outra variável string a partir do endereço-string2. Entre:

```
Ready  120  STRING TESTE
Ready  0  TESTE ASSIGN " 0123456789A "
Ready  0  TESTE 11 TESTE MOVELN
Ready  0  TESTE WRITELN
0123456789A0123456789A
```

13.3.5 A palavra CONCAT

A palavra CONCAT da biblioteca STRING WORDS é usada para concatenar duas variáveis strings. Seu formato é:

endereço-string2 endereço-string1 CONCAT

A palavra CONCAT copia o conteúdo de uma palavra-string1 a partir do endereço-string1 para uma outra palavra-string2 no fim da seqüência de caracteres contida a partir do endereço-string2, concatenando as duas strings em palavra-string2. O conteúdo da palavra-string1 não é alterado por esta instrução. Por exemplo:

```
Ready  20  STRING TESTE1
Ready  30  STRING TESTE2
Ready  0  TESTE1
       ASSIGN " e Animacao Grafica "
Ready  0  TESTE2 ASSIGN " Programacao "
Ready  PAD
       ASSIGN " O Sistema GraFORTH : "
Ready  PAD  0 TESTE2  CONCAT
Ready  PAD  WRITELN
O Sistema GraFORTH : Programacao
Ready  0  TESTE2  0 TESTE1 CONCAT
Ready  0  TESTE2  WRITELN
Programacao e Animacao Grafica
```

13.3.6 A palavra COMPARE

A palavra COMPARE da biblioteca STRING WORDS é usada para comparar os caracteres de duas variáveis strings. Seu formato é:

endereço-string1 endereço-string2 COMPARE

Se a string1 (armazenada na memória a partir do endereço-string1) for maior do que a string2 (armazenada na memória a partir do endereço-string2), então a palavra COMPARE retornará com o valor 1. Se a string1 for menor do que a string2, o valor devolvido pela palavra COMPARE é -1. E se as duas strings forem iguais, a palavra COMPARE retorna com o valor 0.

A comparação feita pela palavra COMPARE é em ordem alfabética e seqüencial. Por exemplo:

```
Ready  ABORT
Ready  STACK
Ready  100  STRING TESTE
Ready  10  TESTE ASSIGN " 123456 "
Ready  PAD ASSIGN " 123455 "
Ready  PAD  10  TESTE  COMPARE
[-1 ]
```

13.4 Verificando o acionamento das teclas

O sistema GraFORTH permite verificar se alguma tecla do seu sistema Apple II foi pressionada ou não durante a execução do seu programa, sem precisar parar o processamento tal como ocorre na execução da palavra GETC. A palavra GETKEY faz leitura da localização de memória do teclado do seu Apple II diretamente.

Quando uma tecla é pressionada, o seu valor em código ASCII do Apple II é armazenado na localização de memória para o teclado. Se uma tecla for pressionada, o número contido nessa localização de memória é sempre maior do que 127.

A palavra GETKEY lê o valor que estiver armazenado nessa localização de memória e armazena-o no topo da pilha de dados.

A palavra CLRKEY força para que essa localização de memória do Apple II sempre contenha um valor menor do que 128. Portanto, a palavra CLRKEY limpa essa localização de memória para que a tecla seja pressionada após uma certa seqüência de execução de palavras.

Por exemplo:

```
FORGET IT
: IT ;
: TECLA
  CLRKEY
  10500 0 DO LOOP
  GETKEY 127 > IF
    CLRKEY
    ABORT
  THEN ;
: PRINCIPAL
  PRINT " IMPRIMINDO OS CARACTERES E "
  SPCE PRINT " SEUS VALORES EM ASCII "
  250 160 DO
    I PUTC SPCE . SPCE SPCE
    TECLA
  LOOP ;
CLOSE RUN
```

13.5 Usando o sistema operacional do GraFORTH

Podemos usar o sistema operacional do GraFORTH, que é compatível com DOS 3.3 (Disk Operating System 3.3) do Apple. Igualmente como ocorre em Applesoft BASIC, podemos passar o controle do GraFORTH ao seu sistema operacional, bastando exibir o caractere ConTRoL-D antes da instrução PRINT. O formato da instrução

que habilita executar os comandos do sistema operacional no GraFORTH é então:

```
CR 132 PUTC PRINT " comando-DOS " CR
```

As palavras CR servem para isolar os blocos de palavras internos, exibindo uma alimentação de linha. A instrução 132 PUTC exibe um caractere ConTRoL-D, o qual instrui o GraFORTH a passar o comando ao sistema operacional. O comando-DOS é um comando no formato exigido pelo sistema operacional DOS 3.3. Por exemplo:

```
Ready CR 132 PUTC PRINT " CATALOG " CR
```

13.6 O programa DOS

A fim de evitar a tarefa tediosa de entrar todo um monte de palavras para passar o controle ao DOS, toda vez que você desejar executar um comando do sistema operacional, elaboramos o seguinte programa:

```
( PROGRAMA : DOS)
FORGET IT
: IT ;
60  STRING AREADOS
0 VARIABLE CONT
: DOS
  132 0 AREADOS POKE
  CR PRINT " Entre o Comando DOS : "
  SPCE 1 AREADOS READLN
  1 AREADOS PEEK 141 = IF
    CONT 1 + -> CONT
  ELSE
    CR 0 AREADOS WRITELN CR
    0 -> CONT
  THEN
  CONT 2 < IF
    DOS
  THEN ;
  CLOSE RUN
```

O programa DOS simula o comando (CTRL)(D) do editor de textos do sistema GraFORTH.

Você pode salvar as palavras definidas no programa DOS junto ao dicionário do sistema GraFORTH, usando a palavra SAVEPRG. Execute a palavra SAVEPRG sem habilitar o modo AUTORUN do sistema. Ver o item 8.6 do Capítulo 8 para maiores detalhes.

13.7 Usando arquivos de textos

Você pode também acessar os dados dos arquivos de textos tal como acontece em Applesoft BASIC. Você deve abrir os arquivos de textos com o próprio comando do DOS. Os dados desse arquivo podem ser lidos ou armazenados, usando as palavras READLN ou WRITELN respectivamente. Por exemplo:

```
( TESTE.LEITURA )
FORGET IT
: IT ;
20 STRING NOME.ARQUIVO
: ABRIR.ARQUIVO.LEITURA
  PRINT " ENTRE O NOME DO ARQUIVO -> "
  SPCE 0 NOME.ARQUIVO READLN
  CR 132 PUTC PRINT " OPEN "
  0 NOME.ARQUIVO WRITELN CR
  CR 132 PUTC PRINT " READ "
  0 NOME.ARQUIVO WRITELN CR ;
: TESTE.LEITURA
  ABRIR.ARQUIVO.LEITURA
  BEGIN
    PAD READLN      ( Lendo uma linha do arquivo. )
    PAD PEEK        ( Comparando o primeiro caractere da
                      linha. )
  170 ()            ( Com o símbolo * . )
  WHILE             ( Testando o fim do arquivo de textos. )
    PAD WRITELN     ( Exibindo a linha do arquivo na tela. )
  REPEAT            ( Volta a executar a palavra BEGIN. )
  CLOSE ;
CLOSE RUN
```

A palavra ABRIR.ARQUIVO.LEITURA abre o arquivo de textos que você desejar, preparando-o para leitura de forma que as palavras READLN executadas a seguir são para obter uma linha do arquivo.

Como o sistema GraFORTH não faz verificação do fim de arquivo e nem possui uma função tal como as instruções ONERR GOTO ou ONERR GOSUB do Applesoft BASIC, vamos entrar o caractere * na primeira posição da linha logo após a última linha, possibilitando a palavra TESTE.LEITURA testar o fim do arquivo. Entre e salve os seguintes textos no editor de textos do sistema GraFORTH:

```
A    ( AUTONUM )
10   TESTE DE LEITURA DO ARQUIVO DE
20   TEXTOS A PARTIR DO DISCO COM O
30   PROGRAMA TESTE.LEITURA.
40   ESTA E A ULTIMA LINHA DO ARQUIVO.
50 * ( * e PRIMEIRO CARACTERE DA LINHA )
```


Para fechar os arquivos de textos, basta entrar a palavra CLOSE do sistema GraFORTH. Como o sistema operacional do GraFORTH só permite um buffer de arquivo, então só um arquivo de textos pode ser aberto por vez.

13.8 Conectando as interfaces do sistema Apple II

Você pode usar o comando IN#n e PR#n (com n variando de 1 a 7) para transferir dados através das interfaces de ou para os periféricos conectados nos slots do seu Apple II. O comando PR#0 não é reconhecido pelo sistema GraFORTH. Portanto, nunca entre PR#0 para desconectar um periférico, mas sim, acione as teclas ⟨CTRL⟩ ⟨RESET⟩.

Por exemplo, se o seu sistema Apple II tiver uma "placa de 80 colunas", você pode executar o comando PR#3, entrando:

```
Ready CR 132 PUTC PRINT " PR#3 " CR
```

13.9 Chamando as rotinas em linguagem de máquina

Você pode chamar uma rotina em linguagem de máquina a partir do sistema GraFORTH através da palavra CALL, cujo formato é:

```
endereço CALL
```

A palavra CALL aguarda, no topo da pilha de dados, um valor; interpreta-o como o endereço contendo uma rotina a executar; e passa o controle a essa rotina.

Podemos então carregar um programa binário (BLOAD) numa área de memória livre e executar através do GraFORTH. Podemos também montar um programa em linguagem de máquina a partir da palavra POKE do dicionário do GraFORTH. Ou podemos executar as palavras do dicionário através de um endereço dado pela palavra ' (apóstrofo), que coloca no topo da pilha o endereço de uma parte da definição da palavra que a segue, por exemplo:

```
Ready ABORT
Ready ' HOME
[-32510 ]
Ready CALL
```

```
( o GraFORTH limpará a tela )

Ready  HEX  6000  DECIMAL
[ 24576 ]
Ready  CALL

( reentra ao sistema GraFORTH )
```

Podemos, antes de chamar (CALL) uma rotina em linguagem de máquina, modificar os valores dos registradores A, X, Y e P, usando as variáveis AREG, XREG, YREG e PREG respectivamente. Por exemplo, podemos simular a função PDL do Applesoft BASIC através de uma nova palavra PDL no sistema GraFORTH:

```
: PDL                ( aguarda valor 0, 1, 2 ou 3 na pilha )
  ->XREG             ( entrando o num. do Paddle no Reg. X )
  64286  CALL        ( chamando a rotina $FB1E do monitor )
  YREG ;             ( empilha o valor do Reg. Y afetado )
```

Se você tiver alguma dúvida sobre o funcionamento da função PDL do Applesoft BASIC, consulte o manual do seu sistema Apple II ou o livro *Guia do programador Applesoft/Integer BASIC* da Hemus Editora.

13.10 Compilando uma tabela de números

A palavra , (vírgula) do dicionário do sistema GraFORTH possibilita gerar uma tabela de valores numéricos na definição de uma palavra do GraFORTH. Por exemplo:

```
: TABELA.DE.CORES
  1 , 2 , 3 , 5 , 6 , 7 , ;
```

A palavra , (vírgula) montará em tempo de compilação uma palavra-tabela no topo do dicionário do sistema GraFORTH, armazenando nela os valores numéricos dados na definição da palavra-tabela. Cada posição útil da palavra-tabela armazena um byte (um valor de 0 a 255). E para referenciar a palavra-tabela, basta usar a palavra ' (apóstrofo) para descobrir qual o endereço da parte da definição da palavra-tabela que contém valores. Experimente:

```
: TABELA
  0 , 1 , 2 , 3 , 5 , 7 , ;
: VER.TAB
  ' TABELA
  BEGIN
    DUP  PEEK  DUP
  96 <> WHILE
    PRINT " NUMERO " SPCE . CR
    1 +
  REPEAT ;
```


Devido à estrutura da compilação de uma palavra do GraFORTH no seu dicionário, você nunca deve começar uma tabela de valores com um número maior do que 127 ou igual a 10, pois o valor 10 é um flag do compilador do sistema GraFORTH, e os valores menores do que 128 são usados para delimitar o fim do nome de cada palavra do dicionário (ver o item 13.11.2 para mais informações).

Portanto, a palavra , (vírgula) deve ser usada somente na definição de palavras e no formato:

```
: palavra-tabela  num1 , num2 , [ num3 . . ., ] ;
```

onde o num1 é um valor menor do que 128 e diferente de 10. A palavra , (vírgula) deve separar cada valor da palavra-tabela e estar separando o último elemento e a palavra ;.

13.11 Analisando o dicionário de palavras

As palavras do dicionário do sistema sem modificações ocupam uma área de memória que compreende os endereços desde $6EB0 a $82CA. Essa área pode ser ampliada, se empilharmos novas palavras no topo do dicionário. O ponteiro (PRGTOP) do dicionário de palavras do sistema GraFORTH sem modificações aponta para o endereço $82CB (esse número depende da versão do GraFORTH), onde está armazenado o endereço da definição da última palavra do dicionário do GraFORTH (CHS).

Existem duas palavras no dicionário que permitem examinar os endereços das palavras e o limite superior do dicionário. São recursos das palavras $LIST e PRGTOP respectivamente.

13.11.0 A palavra $LIST

A palavra $LIST permite listar todo o dicionário de palavras ou uma parte dele. A execução da palavra $LIST é similar à da LIST, com exceção de a palavra $LIST exibir também os endereços das definições compiladas das palavras. Por exemplo:

```
Ready  $LIST
```

```
$82B1  CHS
$82A3  ABS
$8281  SGN
$8251  CALL
$8246  PREG
$823B  YREG
(etc. . .)
```

Os endereços listados aqui podem variar, dependendo do número da versão do GraFORTH.

Vamos agora definir uma palavra IT e listar novamente o dicionário:

```
Ready  : IT ;
Ready  $LIST
$82CB  IT
$82B1  CHS
$82A3  ABS
$8281  SGN
$8251  CALL
$8246  PREG
$823B  YREG
(etc. . .)
```

13.11.1 A palavra PRGTOP

A palavra PRGTOP coloca no topo da pilha de dados o endereço do topo do dicionário de palavras. Por exemplo:

```
Ready  FORGET IT
Ready  $LIST
$82B1  CHS
$82A3  ABS
$8281  SGN
$8251  CALL
$8246  PREG
$823B  YREG
(etc. . .)

Ready  ABORT
Ready  STACK
Ready  PRGTOP
[ -32053 ]
```

Note que o valor decimal -32053 corresponde ao valor hexadecimal $82CB.

13.11.2 A estrutura das palavras do dicionário

Analisadas as palavras $LIST e PRGTOP, agora analisaremos a estrutura das palavras do dicionário.

Vamos compilar uma palavra IT que não fará absolutamente nada:

```
Ready  : IT ;
Ready  $LIST
$82CB  IT
(etc. . .)
Ready  PRGTOP
-32048
Ready  BYE
*
```

Dentro do sistema monitor, vamos listar os códigos hexadecimais armazenados na faixa memória de $82CB a $82D1 (de -32053 a -32047), para analisar como a palavra IT foi compilada e como é a estrutura do dicionário de palavras:

```
*82CB.82D1
82CB-  B1  82  C9  D4  60  CB
82D1-  82
```

Analisando os códigos hexadecimais, podemos afirmar que:

Os endereços $82CB e $82CC armazenam o endereço ($82B1) da palavra CHS, a qual é a penúltima palavra do dicionário logo depois da palavra IT.

Os números C9 e D4 são os valores em código ASCII dos caracteres I e T.

O número logo após os caracteres IT (60) é uma instrução que corresponde ao comando RTS da linguagem Assembly 6502.

Nos endereços $82D0 e 82D1, está armazenado o endereço da palavra IT ($82CB). Neste exemplo dado, a palavra PRGTOP aponta ao endereço $82D0.

Vamos esquecer essa palavra IT e compilar novamente uma outra palavra IT:

```
Ready  FORGET IT
Ready  : IT 1 2 + ;
Ready  $LIST
$82CB  IT
(etc. . .)
Ready  PRGTOP
[ -32035 ]
Ready  BYE
```

```
*82CBL
82CB- B1 82                    ( Aponta p/ o endereco $82B1 )
82CD- C9 D4                    ( Caracteres IT )
82CF- 20 CD 72  JSR $72CD      ( Executa a rotina $72CD p/ )
82D2- 01 00                    ( empilhar o valor 0001 )
82D4- 20 CD 72  JSR $72CD      ( Executa a rotina $72CD p/ )
82D7- 02 00                    ( empilhar o valor 0002 )
82D9- 20 54 7C  JSR $7C54      ( Executa a rotina + )
82DC- 60        RTS
82DD- CB 82                    ( Aponta p/ o endereço $82CB )
```

Podemos resumir a estrutura das palavras do dicionário na figura 13.0.

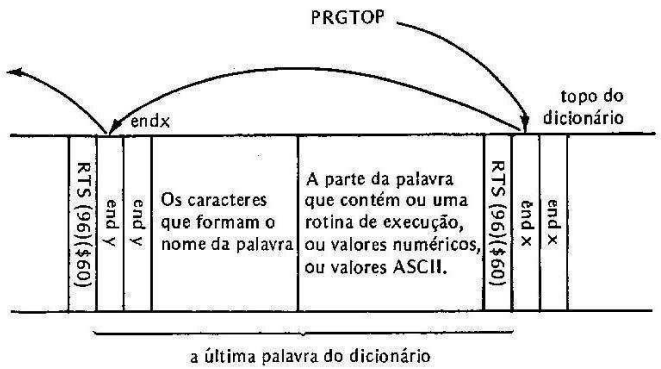


Figura 13.0 — A estrutura das palavras do dicionário.

13.12 Comunicação e segmentação de programas

Conforme já mencionamos, o sistema GraFORTH permite carregar (BLOAD) um programa binário na memória a partir de um disco e executá-lo (CALL). Esta é uma forma de comunicação de programas permitida pelo sistema GraFORTH. Podemos também armazenar dados num arquivo de texto ou binário para posterior processamento por um outro sistema aplicativo ou utilitário. Por exemplo, você pode criar e editar os arquivos de dados no editor de textos do sistema GraFORTH para serem processados pelo Applesoft BASIC. Naturalmente, você pode passar o controle de execução de um programa em GraFORTH para processar um outro também em GraFORTH.

Desse último recurso de comunicação de programas possibilitado pelo sistema GraFORTH, gera-se um outro recurso bastante útil que é o de segmentação de programas; isto significa que você pode partir um programa extremamente grande em dois programas pequenos.

Você pode agora entrar no editor de textos do sistema GraFORTH, carregar (GET) todos os arquivos demonstrativos, listá-los (LIST), analisá-los um a um a fim de entender melhor a lógica de programação em GraFORTH.

No decorrer deste livro ainda vamos descrever todos os recursos aqui abordados.

Parte quinta

Gráficos bidimensionais

14

APRESENTANDO AS CAPACIDADES GRÁFICAS DO GRAFORTH

14.0 A capacidade gráfica do sistema Apple II

Certamente você, usuário de um sistema Apple II (II, II Plus, IIe, IIc, ou enhanced IIe ou compatível), sabe da grande capacidade gráfica que este computador possui. Quem ainda não viu um jogo animado feito para o Apple II e não ficou fascinado com as cores e detalhes das animações? O que alguns não sabem é como o Apple II consegue transformar as seqüências de 1s e 0s em pontos, linhas e cores. É isto que estamos querendo sintetizar para você.

Os sistemas Apple II possuem, pelo menos, os três seguintes modos de resolução:

1. Texto (40 colunas por 24 linhas).
2. Baixa resolução com 16 cores (40 por 48 pixels).
3. Alta resolução com 4 cores (280 por 292 pixels).
   (Essas 4 cores podem ser escolhidas de 2 grupos.)

Nota: A palavra pixel vem da contração de *picture cell* e representa a menor unidade que compõe a tela de resolução, ou seja, ponto na tela de alta resolução.

Vamos analisar somente o modo texto e o modo gráfico de alta resolução que são os utilizados pelo sistema GraFORTH:

– Quatro áreas de memória no espaço de endereçamento do processador são usadas para a geração de imagem. Duas áreas de 1 Kb são reservadas como páginas de texto. Cada byte na página de texto representa um caractere.

– Duas áreas de 8 Kbs são reservadas como páginas de alta resolução, sendo que cada um dos 7 primeiros bits de cada byte representa um pixel na tela. O bit remanescente controla a cor do byte.

As áreas ocupadas por essas páginas de vídeo são listadas no Quadro 14.1.

Quadro 14.1 Áreas ocupadas pelas páginas de vídeo	
Página 1 de textos:	$0400 a $06FF
Página 2 de textos:	$0800 a $0BFF
Página 1 de alta resolução:	$2000 a $3FFF
Página 2 de alta resolução:	$4000 a $5FFF

14.1 O modo texto

A memória de vídeo dos sistemas Apple II não é mapeada linearmente (tal como ocorre nos micros da linha Sinclair ou TRS-80), mas sim de uma forma descontínua em blocos de caracteres que correspondem a regiões alternadas na tela.

A posição de memória $0400 (1024) da página de texto corresponde à posição no canto superior esquerdo da tela no modo texto; sendo que as posições subseqüentes da memória ($0401, $0402 etc.) representam as posições subseqüentes na primeira linha da tela até o canto superior direito. Quando as 40 posições (bytes) dessa linha forem percorridas, a posição subseqüente na memória ($0428) corresponderá à posição mais à esquerda do vídeo, porém, a 8 caracteres abaixo da primeira linha. Após ter completado essa linha (a nona linha da tela), os 40 bytes subseqüentes ($0450 a $0477) corresponderão a uma outra linha de 40 posições a 8 caracteres abaixo dela. Ou seja, os 120 primeiros bytes consecutivos formam as 3 seguintes linhas na tela: a primeira, a nona e a décima-sétima. Além disso, os 8 bytes seguintes a esses 120 primeiros ($478 a $47F) são invisíveis.

Isso permite alinhar o endereço com uma potência de 2 (128) e as linhas se iniciam novamente no endereço $480 (1152), o qual corresponde à primeira posição da segunda linha da tela. Os 120 bytes consecutivos desde o endereço $480 (1152) formam as 3 seguintes linhas na tela: a segunda, a décima e a décima-primeira. Os 8 bytes seguintes são invisíveis. Essa seqüência continua desta maneira até que toda a tela seja preenchida.

14.2 O modo gráfico de alta resolução

Cada página gráfica de alta resolução tem 8.192 bytes de memória reservados para ela. Destes, só 7.680 são utilizados para exibir um pixel na tela. Sobram, assim, 512 bytes formando 64 blocos de 8 bytes que não são utilizados.

O mapeamento da memória da página de alta resolução é similar ao do modo texto: 40 bytes para a primeira linha, pula 1/3 da tela,

40 bytes para a linha 64, pula mais 1/3 da tela, mais 40 bytes para a linha 128, um bloco de 8 bytes sem uso, e mais 40 bytes seguintes para a segunda linha, e assim sucessivamente.

14.3 Formação de um caractere

Cada caractere definido pelo modo texto, na verdade, ocupa um espaço formado por uma matriz de 8 x 7 pixels (8 linhas de alta resolução por 7 colunas), sendo que a oitava linha está em branco para servir de separador de linhas, e a primeira e última colunas também estão em branco para servir de espaço entre duas letras, de modo que elas não apareçam coladas uma à outra. Desta forma, um caractere apresentado na tela é formado por uma matriz de 5 x 7 pixels.

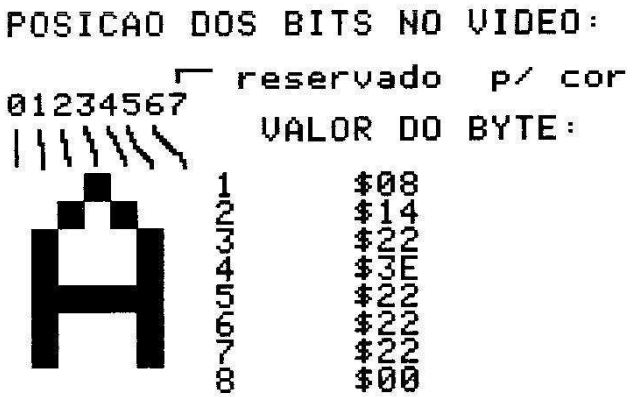


Figura 14.1 — Formação de um caractere.

Se desejar, como exemplo ilustrativo do que explicamos, entre com o seguinte programa que coloca a letra A estilizada na página de alta resolução (Ver a figura 14.1):

```
FORGET IT
: IT ;
: PLOT-A
   HEX                ( passa para a base hexadecimal )
   ERASE              ( apaga a tela no modo grafico )
   08 2000  POKE      ( monta a letra A byte a byte )
   14 2400  POKE
   22 2800  POKE
   3E 2C00  POKE
   22 3000  POKE
```

```
22  3400  POKE
22  3800  POKE
00  3C00  POKE
18  VTAB          ( posiciona na 18a. linha da tela )
DECIMAL ;
```

Ao entrar com a palavra PLOT-A, você verá a letra A no canto esquerdo superior. Os valores dos bytes que formam os caracteres ASCII na tela de texto estão armazenados numa área de memória ROM de 1.024 bytes. O hardware do Apple II lê continuamente os valores nesta área para exibir o caractere apropriado no vídeo.

O Apple II permite que se ligue ou desligue qualquer um dos 53.760 pontos da tela gráfica individualmente, bastando localizar a posição de memória correspondente à posição desejada na tela e setar (zerar) ou ressetar (colocar 1) no bit indicado.

É possível alternar rapidamente a página de alta resolução que é exibida entre as duas existentes – que funcionam como verdadeiros buffers de figuras – possibilitando efeitos de animação (o sistema GraFORTH faz isso automaticamente na apresentação de imagens tridimensionais).

O tipo de mapeamento das linhas nas páginas de alta resolução é extremamente complexo de se manipular através de uma visão do tipo "posição x,y", onde você acessaria diretamente uma posição da tela. O sistema GraFORTH se apresenta como um editor de figuras e desenhos extremamente complexo que facilita sobremaneira a sua manipulação desse emaranhado de seqüências de bytes, com uma quantidade de recursos infinitamente superiores aos de qualquer versão da linguagem BASIC disponível.

14.4 As capacidades gráficas do GraFORTH

O sistema GraFORTH permite a utilização de dois modos de resolução: o modo texto pelo uso da palavra TEXT e o modo gráfico que é o modo usual e possibilitando usar as duas páginas de alta resolução. Uma vez no modo texto, podemos retornar ao modo gráfico utilizando a palavra GR que reestabelece a saída normal do GraFORTH e posiciona o modo gráfico.

Para referenciar às posições na tela no modo gráfico, o sistema GraFORTH utiliza as coordenadas cartesianas, sendo a origem (x=0 e y=0) localizada no canto superior esquerdo. A coordenada x varia de 0 até 255 (da esquerda à direita) e a coordenada y varia de 0 a 191 (de cima para baixo), lembrando sempre que a coordenada x corresponde à coluna e a y corresponde à linha de um determinado ponto.

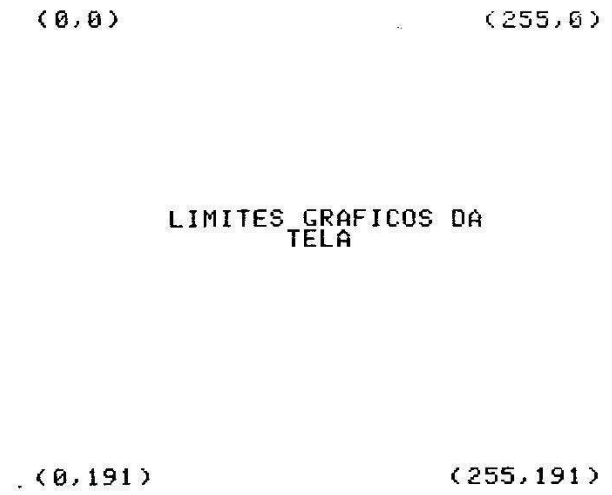


Figura 14.2 – Os limites da tela de alta resolução.

O leitor mais atento deve ter notado que existe uma discrepância entre o número de colunas que o sistema GraFORTH realmente apresenta e o número possível de ser acessado pelo hardware do seu sistema Apple II. De fato, o GraFORTH utiliza-se apenas das 256 primeiras colunas (ao invés das 280 colunas possíveis) pelo fato desse número poder ser armazenado num único byte – ao contrário do número 280 que necessita de 2 bytes – além de proporcionar uma possibilidade de acelerar o processo de plotagem de linhas ou pontos.

Essa velocidade paga o preço de possuirmos uma tela gráfica de tamanho 9% menor.

Uma grande vantagem que o sistema GraFORTH permite é a de manipular textos e gráficos concomitantemente numa mesma tela, o que não é muito fácil com o Applesoft BASIC, sentindo-se os usuários do Apple II no mínimo aborrecidos por essa desvantagem.

Como ele consegue essa magia? Ela é simples. Os caracteres, em lugar de serem colocados no vídeo pelo hardware do sistema Apple II, são literalmente desenhados na página de alta resolução, byte a byte, assim como fizemos no exemplo anterior com a letra A. Simples não? Deixamos de lado a manipulação de caracteres que será vista com mais detalhes na parte sexta.

15

PALAVRAS DE MANIPULAÇÃO DE GRÁFICOS BIDIMENSIONAIS

15.0 Palavras de controle gráfico

Efetivamente, num computador, os gráficos, desenhos e figuras são manipulados nos termos dos planos do vídeo, de modo que só existem duas dimensões: linhas e colunas.

É dessa forma que classificamos as palavras do GraFORTH em bidimensionais quando envolvem basicamente duas coordenadas (x e y) e tridimensionais quando uma terceira é acrescentada (z). O sistema GraFORTH simula esta terceira dimensão numa tela bidimensional.

Costuma-se, dentro do campo da computação gráfica, chamar de "primitivas" as rotinas bidimensionais de manipulação do vídeo, pois em cima delas são fabricadas as rotinas tridimensionais.

Destas rotinas, a mais básica pode ser considerada a que plota pontos. A partir desta, podem ser construídas rotinas para traçar linhas, círculos, curvas mais complexas, além de preenchimento de figuras.

15.1 A palavra PLOT

Para ilustrar a utilização destas palavras gráficas, devemos antes preparar a tela, já que não queremos que o texto já existente fique sobreposto ao gráfico. Para isso, o GraFORTH possui uma palavra que estabelece janelas de texto. Assim, vamos reservar as 6 últimas linhas da tela para o texto, deixando livre o resto da tela (note que isso não impede que tracemos alguma reta ou desenho sobre essa janela de textos).

```
Ready  0 40 18 23 WINDOW
Ready  ERASE
```

Com estas palavras, limpamos toda a tela e podemos agora trabalhar com gráficos. Iniciamos plotando um ponto na tela. Para tanto,

necessitamos especificar onde ele será plotado, ou seja, as coordenadas x e y. Seguindo a notação polonesa inversa, da qual o FORTH se utiliza, a palavra PLOT tem a seguinte forma:

```
coordenada-x  coordenada-y  PLOT
```

a significar que a palavra PLOT espera 2 valores na pilha, onde no topo da pilha deve estar a coordenada y e no segundo elemento da pilha o valor da coordenada x.

Podemos, como exemplo, confirmar onde está a origem do sistema, entrando:

```
Ready  0 0 PLOT
```

Conforme mencionamos anteriormente, o valor máximo das coordenadas é 255 e 191, para x e y respectivamente. Podemos verificar onde está este ponto, entrando:

```
Ready  255 191 PLOT
```

Este ponto está localizado sobre a nossa janela de texto e, portanto, texto e gráficos podem conviver simultaneamente numa mesma tela (o que pode ocorrer é que o ponto seja apagado ao ser dado um scroll na janela de texto).

15.2 A palavra LINE

Para traçar um segmento de reta, precisamos de dois pontos de referência: o ponto inicial e o final da reta. No sistema GraFORTH, o último ponto utilizado por uma palavra gráfica fica armazenado na página zero da memória do Apple II. Mais precisamente, na posição $80 está a última coordenada x e em $82 está a última coordenada y.

Podemos verificar o fato entrando:

```
Ready  255  191  PLOT          ( plota um ponto na tela )
Ready  HEX                     ( passa para a base hexadecimal )
Ready  82  PEEK  80  PEEK      ( coloca y e x na pilha )
Ready  DECIMAL                 ( volta para a base decimal )
Ready  .  CR  .                ( exibir os valores x e y )
```

O resultado na tela são os valores 255 e 191 correspondentes ao último ponto que entramos como exemplo.

Desta forma, a palavra LINE, para traçar segmentos de reta, se utiliza destas posições de memória como sendo seu ponto inicial e

espera na pilha as coordenadas do ponto final, na mesma notação que a palavra PLOT usa. Vamos aproveitar este último ponto que usamos e traçar uma diagonal cortando toda a tela:

```
Ready  0  0  LINE
( observe que a parte sobre o texto )
( fica encoberta e/ou apagada )
```

15.3 A palavra POSN

Se quisermos, inicialmente, posicionar o ponto inicial de uma reta sem plotá-lo, podemos fazê-lo usando a palavra POSN:

```
Ready  125  10  POSN
Ready  10  125  LINE
```

Observe que a palavra POSN simplesmente atualiza os bytes $80 e $82, para posterior uso de outras palavras, sem realizar nenhuma tarefa, usando a mesma notação de PLOT e LINE, ou seja:

```
coordenada-x  coordenada-y  POSN
```

15.4 Exercício de aplicação

Utilizando nosso conhecimento dessas três palavras já podemos desenvolver um programa. Por exemplo, desenhar um retângulo utilizando como dados dois pontos quaisquer que representem dois extremos de uma das diagonais do retângulo, uma destas extremidades sendo o último ponto plotado e a outra sendo passada pela pilha.

O retângulo é posicionado na horizontal. Chamando de LX e LY as coordenadas do último ponto plotado, e de X e Y as coordenadas passadas pela pilha, temos, conforme a figura 15.0, a seguinte configuração esquemática dos vértices deste retângulo.

Qualquer uma dessas configurações pode ser desenhada, utilizando-se as seguintes seqüências (assumindo que LX e LY sejam as coordenadas do último ponto plotado):

```
LX   Y  LINE
 X   Y  LINE
 X  LY  LINE
LX  LY  LINE
```

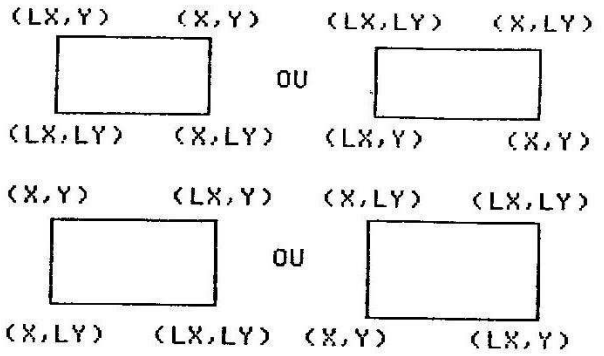



Figura 15.0 – Vértices do retângulo.

A maneira mais simples de montar este programa é guardar os valores dos vértices em variáveis e chamá-los conforme sejam parâmetros necessários. Desta forma, podemos montar o programa:

```
Ready  VARIABLE  X
Ready  VARIABLE  Y
Ready  VARIABLE  LX
Ready  VARIABLE  LY
Ready  : RETANGULO       ( nome do nosso programa )
Ready  ->Y  ->X          ( os valores da pilha são armazenados )
Ready  128 PEEK -> LX    ( os valores do último ponto são )
Ready  130 PEEK -> LY    ( armazenados também )
Ready  LX  Y  LINE       ( traçam as linhas que formam )
Ready  X   Y  LINE       ( o retângulo )
Ready  X  LY  LINE
Ready  LX LY  LINE ;
```

Podemos agora testar este nosso programa. Que tal usá-lo para emoldurar uma janela de texto? Entre, então, o seguinte:

```
Ready  2  30  10  15  WINDOW
Ready  ERASE
Ready  12  78  POSN
Ready  225  130  RETANGULO
```

Gostou? Agora, você já tem bem visualizado onde está a janela de texto, sendo que a moldura está fora dessa janela. Mas, o que acontece se você colocar um valor fora dos limites estipulados na tela?

A primeira reação que você tem ao responder essa pergunta é dizer que o sistema exibirá alguma mensagem de erro e parará a execução do seu programa. Contudo, isso não ocorrerá. O sistema Gra-

FORTH necessita ser o mais rápido possível, e um teste de verificação tornaria este processo mais lento (ver o item 12.10 do Capítulo 12).

Assim, o GraFORTH simplesmente trabalhará com o valor excedente. Por exemplo, se for dado o valor 300 para a coordenada X, ele plotará na coluna correspondente ao valor excedente que é 44 (= 300 – 256).

15.5 Desenhando círculos em GraFORTH

Já sabemos como plotar um ponto na tela, como traçar segmentos de reta e como desenhar retângulos. Vamos então desenvolver mais uma rotina para tratamentos de gráficos: um programa para traçar círculos. Precisamos, assim, analisar o seguinte problema:

Consideremos, por simplicidade, a equação matemática de um círculo genérico centrado na origem

$$X^2 + Y^2 = R^2,$$

Isolando o valor Y, obtemos:

$$Y = +\sqrt{R^2 - X^2}$$

Desta forma, para desenhar um quarto de círculo, podemos incrementar x desde 0 até R em passos unitários, resolvendo a equação para y positivo a cada passo (os outros quartos de círculo podem ser desenhados por simetria). Esse sistema funcionaria em Applesoft BASIC, porém é impraticável por necessitar da extração de raiz e operação com ponto flutuante, o que não é possível com o GraFORTH. Esta fórmula também apresenta o desagrado de formar grandes lacunas no círculo quando o valor de x se aproxima do valor de R, pois a tangente do círculo tende a infinito, quando o y tende a R, ou em outras palavras, o y decresce muito mais do que uma unidade. Veja a figura 15.1 para entender o que aconteceria:

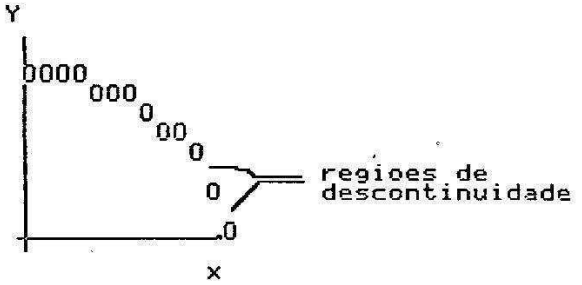


Figura 15.1 – Pontos de descontinuidade de um círculo.

Outro método que também se torna ineficiente e extremamente lento, como o anterior, seria o de plotarmos os pontos (RcosA, RsenA) incrementando o ângulo A com valores de 0 a 90.

Devemos estar sempre atentos para promover essa plotagem de círculo o mais veloz possível; procuraremos então conseguir um processo que seja o mais simples e necessite de menos funções e em que possamos trocar palavras mais lentas por outras mais rápidas.

Uma forma de acelerar o processo é diminuir o número de cálculos e utilizar a simetria dos pontos da circunferência, sendo necessário, para isso, que o centro esteja na origem, simplesmente trocando x com y ou alterando o seu sinal, como mostrado abaixo:

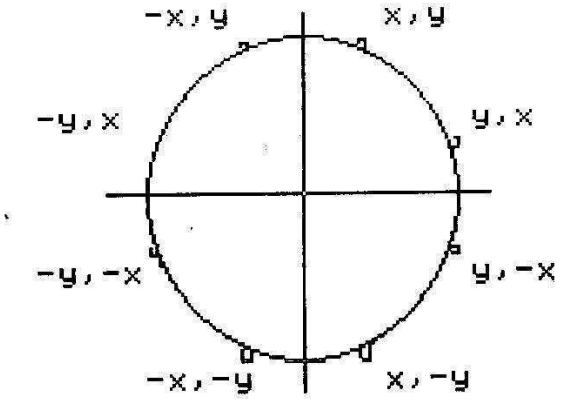


Figura 15.2 – Pontos de simetria do círculo.

Assim, calculando um único ponto, podemos facilmente plotar este e mais sete pontos sem nenhum cálculo. Podemos então implementar uma palavra no GraFORTH que plote os oito pontos, sendo dados apenas um deles, armazenado pelas suas coordenadas x e y (edite seu programa através do editor, prevendo futuras alterações):

```
A    ( AUTONUM )
10 :  PLTCIR
20     X Y PLOT X CHS Y CHS PLOT
30     Y X PLOT Y CHS X CHS PLOT
40     Y X CHS PLOT Y CHS X PLOT
50     X Y CHS PLOT X CHS Y PLOT ;
```

Este programa só funcionará corretamente para uma circunferência centrada na origem. Para uma circunferência centrada num ponto genérico, devemos acrescentar as coordenadas do centro especificado

(vamos assumir que esteja nas coordenadas LX e LY) antes de cada ponto ser plotado. Basta, então, alterar a palavra PLOT criando uma nova palavra que vamos definir para fazer esta adição:

```
5  : CPLOT  LY + SWAP  LX + SWAP  PLOT ;
```

Como foi dito, devemos tornar este programa de círculo o mais rápido possível. Podemos acelerar o programa PLTCIR, usando a vantagem da pilha de dados, cujas manipulações são mais rápidas que as chamadas de variáveis. Nesta nova implementação, você pode notar entre os parênteses como fica a pilha após cada linha de instruções, notando também como as coordenadas necessárias são montadas todas de uma vez e depois é chamada a rotina de plotagem oito vezes (utilize o editor para alterar os valores):

```
20   : PTLCIR  X  Y          ( X Y )
30     OVER OVER CHS         ( X Y X -Y )
40     OVER CHS OVER         ( X Y X -Y -X -Y )
50     OVER OVER CHS         ( X Y X -Y -X -Y -X Y )
60     OVER OVER SWAP        ( X Y X -Y -X -Y -X Y Y -X )
70     OVER OVER CHS         ( X Y X -Y -X -Y -X Y Y -X Y X )
80     OVER CHS OVER         ( X Y X -Y -X -Y -X Y Y -X Y
                               X -Y X )
90     OVER OVER CHS         ( X Y X -Y -X -Y -X Y Y -X Y
                               X -Y X -Y -X )
100    CPLOT CPLOT CPLOT CPLOT CPLOT CPLOT CPLOT CPLOT ;
```

Para usar este programa, devemos calcular os valores X e Y através de um outro programa.

Esta rotina foi desenvolvida por Bresenham, sendo conhecida como gerador de circunferências incremental. É chamado de incremental por gerar os pontos através de adições nas coordenadas x e y exclusivamente. A seguir, apresentamos o algoritmo de Bresenham, que supõe o centro do círculo na origem:

```
início
    X = 0
    Y = RAIO
    D = 3 – 2 * – –>
    enquanto X < Y faca
          plote os pontos (PLTCIR)
          se D < 0
              entao
                   D = D + 4 * X + 6
```



```
              senao
                   D = D + 4 * ( X – Y ) + 10
                   Y = Y – 1
          fim-se
          X = X + 1
    fim-enquanto
    se X = Y
        entao
             plote os pontos (PLTCIR)
    fim-se
fim
```

Como se chega a esse algoritmo não faz parte do nosso assunto e quem tiver maior interesse pode consultar o livro *Fundamentals of interative computer graphics*, Foley e Van Dan.

Vejamos como fica esse algoritmo no GraFORTH. Observe bem para ver se você consegue descobrir as mágicas que foram feitas para encurtar a rotina e, conseqüentemente, acelerá-la:

```
0 –> X
RAIO –> Y
3  2  RAIO  *  –  –> D
BEGIN
     PLTCIR
     D  X  4  *  +  6  +  –> D
     D  0  )=  IF
          D  Y  1  –  DUP  –> Y
          4  *  –  –> D
     THEN
     X  1  +  –> X
X  Y  >
UNTIL
```

Se você não consegue notar as semelhanças do algoritmo de Bresenham com o deste programa em GraFORTH, aconselhamos tentar executar, passo a passo, manualmente, as partes onde há dificuldade e exibir o conteúdo da pilha com a palavra STACK para examinar como ela está sendo alterada.

Necessitamos então definir a variável D:

```
Ready  VARIABLE D
```

Para finalizar o nosso programa, precisamos inicializar as variáveis LX e LY que conterão as coordenadas do centro da circunferência

(lembre-se de defini-las caso você não tenha executado o exemplo RETANGULO) e lembrando que o raio do círculo é passado pela pilha:

```
110  : CIRCLE
120    128  PEEK   -> LX
130    130  PEEK   -> LY
140    0 -> X  DUP  -> Y
150    2  *  3 SWAP  -  -> D
160    BEGIN
170        PLTCIR
180        D  X  4  *  +  6  +  -> D
190        D  0  )=  IF
200              D  Y  1  -  DUP  -> Y
210              4  *  -  -> D
220              THEN
230              X  1  +  -> X
240    X  Y  >
250    UNTIL ;
```

Vejamos um exemplo de utilização (lembre-se de manter uma janela de texto para não interferir com o desenho):

```
: CIRO5
  ERASE                ( limpa a tela )
  120  50  POSN        ( centraliza o primeiro círculo )
  5                    ( coloca o primeiro raio na pilha )
  5  0  DO             ( comando de loop: repete 5 vezes )
     DUP  CIRCLE       ( preserva o raio, desenha o círculo )
     2  *              ( duplica o raio que estava na pilha )
  LOOP                 ( volta para desenhar outro círculo )
  DROP ;               ( apaga o raio que sobrou na pilha)
```

Você verá surgir na tela 5 círculos de raios em progressão, sendo um o dobro do outro, utilizando como centro o último ponto plotado do círculo anterior.

Para obter uma listagem definitiva, acrescente as linhas de definição das variáveis:

```
5   VARIABLE LX
15  VARIABLE LY
25  VARIABLE X
35  VARIABLE Y
45  VARIABLE D
```

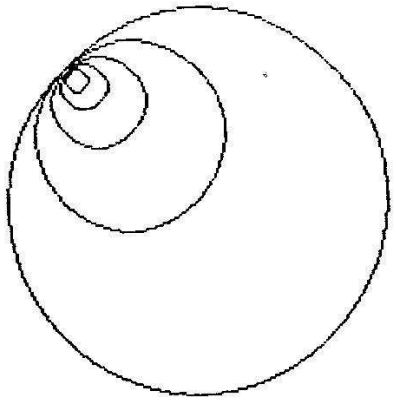



Figura 15.3 – Círculos em progressão.

15.6 A palavra COLOR

Uma das grandes vantagens apresentada pelo sistema Apple II em comparação com outros microcomputadores é a possibilidade de trabalhar com cores. O sistema GraFORTH também pode utilizar esses recursos através da palavra COLOR, que seleciona uma cor dentre o conjunto especificado no Quadro 15.1.

Quadro 15.1 Tabela das cores

0	não usado	
1	verde (1)	
2	violeta (1)	
3	branco (1)	
4	não usado	
5	laranja (2)	(depende do monitor)
6	azul (2)	(depende do monitor)
7	branco (2)	

A sintaxe da palavra é a seguinte:

número-da-cor COLOR

ou seja, a palavra COLOR espera, na pilha, o número que especifica a cor escolhida, segundo o Quadro 15.1. Todas as palavras gráficas

utilizarão essa cor, até que outra seja especificada. Note que existem nessa tabela números entre parênteses que servem para especificar o grupo ao qual pertence a cor. Já que o Apple II só permite 4 cores conjuntas no modo de alta resolução, é aconselhável só utilizar cores de um mesmo grupo (1 ou 2) numa mesma tela.

Devemos frisar que os números 0 e 4 não são usados pelo GraFORTH, e seu uso pode incorrer em sérios danos ao sistema, inclusive perdendo o seu controle.

Para exemplificar, podemos traçar algumas linhas em várias cores:

```
Ready  ERASE
Ready  0  50  POSN
Ready  1   COLOR   40    0   LINE
Ready  2   COLOR   80   50   LINE
Ready  3   COLOR  120    0   LINE
Ready  5   COLOR  160   50   LINE
Ready  6   COLOR  200    0   LINE
Ready  7   COLOR  240   50   LINE
```

Se você está usando um monitor colorido, poderá observar as seguintes cores: verde, violeta, branco, laranja, azul e branco.

Se você não alterar o valor da cor, todas as próximas linhas e pontos serão plotados com a cor branca (número 7), que foi a última cor especificada.

Em monitores monocromáticos ocorre que, para cores diferentes de 3 e 7 (branco), as retas são descontínuas, devido a uma característica de hardware.

15.7 A palavra FILL

Podemos preencher uma área retangular com pontos, na cor especificada, rapidamente, usando a palavra FILL. Ela também espera na pilha as coordenadas x e y do ponto que representa um dos cantos do retângulo, sendo o outro canto oposto a este especificado pela última palavra gráfica do GraFORTH, assim como na sintaxe da palavra RETANGULO, a qual nós implementamos.

```
Ready  1  COLOR          ( seleciona a cor )
Ready  120  125  POSN    ( posiciona canto inferior esquerdo )
Ready  200   75  FILL    ( posiciona canto superior direito e )
                         ( preenche o retângulo determinado )
```

Lembre-se, também, que a palavra FILL atualiza os bytes $80 e $82, ou seja, o ponto especificado na instrução FILL passa a ser o último ponto plotado, usado por uma posterior LINE ou FILL.

Note que ao trabalhar com cores diferentes e que sejam sobrepostas, seus valores são alterados por causa do modo como o hardware do Apple trata as cores.

Por exemplo, preencha uma área com a cor laranja e desenhe uma linha com a cor azul que passe sobre essa área:

```
Ready  5 COLOR   0 0 PLOT   100 100 FILL
Ready  6 COLOR   0 0 PLOT   100 100 LINE
```

Interessante, a cor tornou-se branca! Isso acontece porque usamos cores do mesmo grupo (grupo 2). Tentemos com cores de grupos diferentes:

```
Ready  ERASE
Ready  5 COLOR   0 0 PLOT   100 100 FILL
Ready  1 COLOR   0 0 PLOT   100 100 LINE
```

Veja o resultado! Apareceu uma série de retângulos verdes ao longo da diagonal. Para evitar que isso ocorra, recomenda-se não misturar cores de grupos diferentes quando linhas ou áreas são sobrepostas.

15.8 A palavra UNPLOT

Da mesma maneira que podemos traçar linhas, plotar pontos e preencher áreas, podemos apagá-las separadamente pelas instruções correspondentes.

Para apagar um ponto, portanto, usamos a palavra UNPLOT. Para exemplificar, plotemos um ponto e depois o apaguemos:

```
Ready  ERASE
Ready  2  COLOR
Ready  100  25  PLOT
Ready  100  25  UNPLOT
```

Observe que somente será apagado o ponto se UNPLOT for usado com a mesma cor especificada na hora em que foi plotado. Se uma cor diferente for especificada, a palavra UNPLOT não apagará o ponto, mas colocará uma cor complementar. Utilize este exemplo e veja o que ocorre:

```
Ready  ERASE
Ready  2  COLOR
Ready  25  75  PLOT
Ready  5  COLOR
Ready  25  75  UNPLOT
```

15.9 A palavra UNLINE

Similarmente, existe a palavra UNLINE que pode apagar linhas. Ela possui a mesma sintaxe da palavra LINE e todas as observações feitas para a palavra UNPLOT sobre as cores são igualmente válidas. Vejamos UNLINE em funcionamento:

```
Ready 0 70 PLOT 255 0 LINE
Ready 0 70 UNLINE          ( note que o ponto inicial não )
Ready 255 0 UNPLOT         ( foi apagado, logo devemos )
                           ( executar a operação inversa )
                           ( sob o risco de não apagar tudo )
```

15.10 A palavra EMPTY

Do mesmo modo, as áreas criadas pela palavra FILL podem ser apagadas pela palavra EMPTY. Da mesma forma, as observações sobre a manutenção da cor são mantidas. Exemplo:

```
Ready 175 25 PLOT    225 100 FILL
Ready 175 25 UNPLOT  225 100 EMPTY
```

15.11 Os modos INVERSE e NORMAL

Até agora, você tem escrito os caracteres no modo NORMAL, ou seja, caracteres brancos no fundo preto. Porém, você pode escrever esses caracteres no modo inverso com a palavra INVERSE:

```
Ready ERASE
Ready INVERSE
```

Note que o prompt "Ready" é agora exibido com a cor preta e o plano de fundo está com a cor branca, ou seja, o prompt foi exibido no modo inverso. Desde que ela foi a última palavra mostrada após a execução da palavra INVERSE, ela é a única em modo inverso. Note que INVERSE simplesmente desenha no complemento da cor selecionada, ou seja, preto complementa branco, verde complementa violeta, azul complementa laranja etc.

```
Ready HOME
```

Como a palavra HOME serve para limpar a janela de texto, tudo que estiver nela estará no modo inverso.

Agora, utilize:

```
Ready ERASE
```

Note neste instante que toda a tela está branca. Vamos preencher umas áreas para ver o efeito do modo inverso sobre as cores.

```
Ready 0 0 POSN
Ready 1 COLOR  20  30 FILL
Ready 2 COLOR  40  60 FILL
Ready 3 COLOR  60  90 FILL
Ready 5 COLOR  80 120 FILL
Ready 6 COLOR 100 150 FILL
Ready 7 COLOR 120 180 FILL
```

Veja que as cores foram alteradas pelos seus complementos, ou seja, os retângulos formados, da esquerda para a direita, apresentam as cores violeta, verde, preto, azul, laranja e preto.

Para restabelecer o modo normal, em que estávamos antes, basta entrar:

```
Ready NORMAL ERASE
```

A tela se apresentará novamente escura, com o prompt escrito em branco sobre preto.

15.12 ORMODE e EXMODE

Existem, no sistema GraFORTH, dois modos de gráficos: ORMODE e EXMODE. ORMODE é o modo default, isto é, aquele em que o sistema se encontra quando é inicializado e que não é alterado até que o outro modo seja especificado. ORMODE segue a filosofia de que se algo deve ser plotado na tela, então, deve-se esquecer se já existia algo plotado ou não, e sempre plotar (ou apagar se a palavra for do tipo UNPLOT). Já no modo EXMODE há a seguinte consideração: se eu tenho de plotar um determinado ponto e ele não está plotado, então eu ploto. Se, entretanto, este ponto já está plotado, eu o apago.

Vejamos alguns exemplos para ilustrar melhor o que ocorre com estes dois modos:

```
Ready 50 20 POSN 150  80 FILL
Ready  0  0 POSN 200 120 LINE
```

Este exemplo serve para mostrar o funcionamento do ORMODE, que é o modo no qual nos encontramos. Note que traçamos uma linha sobre a área preenchida. Quando tentamos apagar a linha:

```
Ready  0  0  POSN  200  120  UNLINE
```

observamos que ocorre um corte no retângulo, pois, no modo ORMODE, tudo que é mandado apagar é apagado independentemente do que já estava no vídeo.

Tentemos o mesmo exemplo em EXMODE para ver a diferença:

```
Ready  ERASE  EXMODE
Ready  50  20  POSN  150   80  FILL
Ready   0   0  POSN  200  120  LINE
```

Agora, a linha, ao passar pelo retângulo, apaga os pontos ao invés de acendê-los (pois já estavam acesos) e, fora desta área, plota os pontos (pois estavam apagados). O mesmo ocorre se você desenhar algo sobre um texto — a parte que se sobrepuser ao texto ficará invertida.

Para apagar a linha no modo EXMODE, basta tentar desenhá-la novamente:

```
Ready  0  0  POSN  200  120  LINE
```

o que restabelece o retângulo, já que se processa o inverso do que foi feito quando se traçou a linha — o que estava apagado acendeu e o que estava aceso apagou.

Note que EXMODE funciona igualmente em cores, alterando-as para seus complementos, podendo ser combinado com NORMAL e INVERSE para obter efeitos e combinações diversas.

Podemos, também, em EXMODE, usar as palavras que definimos anteriormente (RETANGULO e CIRCLE) para examinar outros efeitos.

```
Ready  10  10  POSN
Ready  10  10  POSN  50  30  RETANGULO
Ready  30  20  POSN  15  CIRCLE
```

Observamos que os pontos de intersecção entre o círculo e o retângulo e alguns outros pontos estão apagados. Porém, se desenharmos novamente o retângulo, poderemos restituir o círculo:

```
Ready  10  10  POSN  50  30  RETANGULO
```


Devemos nos lembrar (concluindo nossas observações) que se plotarmos um ponto um número par de vezes, ele ficará apagado, e se plotarmos um número ímpar de vezes, ele permanecerá aceso. Use o seguinte programa para verificação:

```
Ready  ERASE
Ready  :  PONTO
Ready     0  DO
Ready     100  100  PLOT
Ready     LOOP ;
```

O número de vezes que o ponto (100,100) será plotado é esperado na pilha. Se você usar um número par de vezes, ele continuará apagado. Se for ímpar, ele acenderá.

15.13 A palavra GPEEK

Você tem agora, dentro dos seus programas, a possibilidade de verificar se algum ponto na tela está aceso ou não. Isto é possível pelo uso da palavra GPEEK, de utilização simples e que reflete a mesma sintaxe das palavras anteriores, ou seja, espera na pilha as coordenadas x e y do ponto a ser testado. Se este ponto está aceso, então a palavra GPEEK deixa na pilha um valor diferente de zero, ou se o ponto estiver apagado, empilha o valor zero. Por exemplo:

```
Ready  100  100  PLOT
Ready  100  100  GPEEK .
Ready   99   99  GPEEK .
```

Ao usarmos esta palavra, devemos ter cuidado, pois ela altera as coordenadas do último ponto plotado assim como as palavras PLOT, UNPLOT, LINE e UNLINE. Confirme examinando o conteúdo das posições de memória $80 (128) e $82 (130):

```
Ready  0  0  POSN  100  100  LINE
Ready  77  77  GPEEK
Ready  128  PEEK . SPCE  130  PEEK .
```

O resultado na tela é 77 77, os valores do último GPEEK.

15.14 Pintando todas as figuras bidimensionais

Podemos com a palavra GPEEK desenvolver mais uma rotina que nos auxiliará bastante. É uma versão melhorada da palavra FILL que

preenche uma região delimitada por pontos já plotados de qualquer forma, não se restringindo a um retângulo. Pode ser uma circunferência, um triângulo ou qualquer outra região mais complexa. Este procedimento é simples, mas altamente recursivo e utiliza a pilha de dados, podendo inclusive causar um *stack overflow*, ou seja, ultrapassar a capacidade da pilha (porém, isso é pouco provável com o uso de figuras comuns).

O procedimento do algoritmo é o seguinte: são armazenadas na pilha as coordenadas do ponto inicial. A partir deste ponto se faz uma busca até encontrar o ponto da linha que esteja mais à direita dentro de uma região. Depois, é feita uma busca na linha acima, onde se procura determinar um ponto que não esteja plotado e que possua um à direita que já esteja plotado (com exceção do primeiro que é localizado apenas se não estiver plotado). O mesmo é feito com a linha abaixo. Os pontos que foram localizados por essa busca são deixados na pilha, os quais serão usados como pontos iniciais no reinício do processo. A linha inicial é plotada desde o ponto inicial até o último ainda aceso. Quando a pilha estiver vazia, o algoritmo está terminado.

A figura 15.4 tenta explicar e elucidar melhor o mecanismo. Na figura (a), a linha que contém o ponto inicial é traçada e as posições dos pontos numerados são armazenadas na pilha. Os números indicam a ordem com que eles aparecem na pilha: o número 1 indica o fundo da pilha e é processado por último. Mostramos apenas uma parte do processo de preenchimento da região e esperamos que o leitor termine a seqüência para melhor entender a sua lógica.

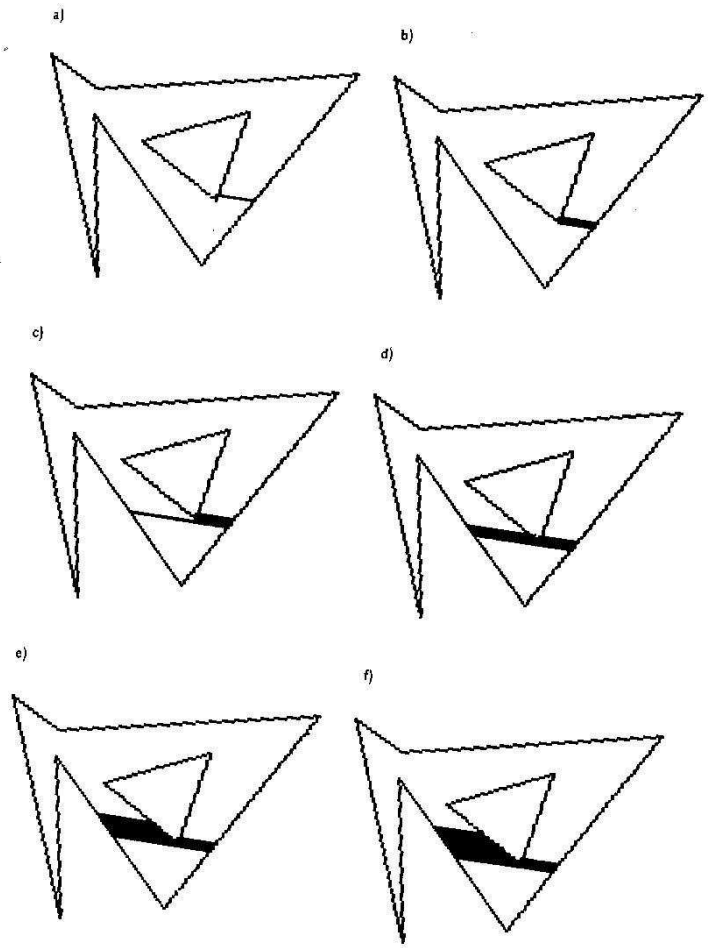


Figura 15.4 – O funcionamento da palavra AREAFILL.

O programa básico, representando o algoritmo acima descrito, pode ser escrito no GraFORTH como segue:

```
VARIABLE X
VARIABLE Y
: FILLING -> Y -> X
  BEGIN
        X 1 + Y GPEEK 0 =
  WHILE
        X 1 + -> X
  REPEAT
  X Y PUSH PUSH
  X Y 1 +
  OVER OVER GPEEK IF
        DROP DROP
  THEN
  X Y 1 -
  OVER OVER GPEEK IF
        DROP DROP
  THEN
  BEGIN
        X 1 - Y GPEEK 0 =
  WHILE
        X 1 - -> X
        X 1 + Y 1 + GPEEK IF
            X Y 1 +
            OVER OVER GPEEK IF
                DROP DROP
            THEN
        THEN
        X 1 + Y 1 - GPEEK IF
            X Y 1 -
            OVER OVER GPEEK IF
                DROP DROP
            THEN
        THEN
  REPEAT
  PULL PULL PLOT X Y LINE ;
: ARFILL
  PUSH PUSH -1 PULL PULL
  BEGIN
        FILLING
  DUP 0 <
  UNTIL
  DROP ;
```


A palavra usada para preencher uma área da tela é ARFILL que espera na pilha as coordenadas x e y de um ponto qualquer interior à região.

Vejamos um exemplo de sua utilização. Desenhemos um círculo e um retângulo interno a esse círculo e preenchamos a região delimitada entre o retângulo e o círculo:

```
Ready ERASE 100 100 POSN 100 CIRCLE
Ready 50 50 PLOT 150 150 RETANGULO
Ready 25 100 ARFILL
```

É importante ressaltar que o programa apresentado não faz testes de limite da tela, o que significa que se você der um ponto que não seja interno a nenhuma região delimitada e fechada, o programa ficará procurando um lado — que não existe — para terminar a busca. Se isso ocorrer, a única forma de sair do looping perpétuo é reiniciar o sistema com as teclas ⟨CTRL⟩⟨RESET⟩ (ou a tecla ⟨RESET⟩, dependendo do seu sistema Apple II).

Aconselhamos, como exercício de fixação dos conceitos, tentar minimizar a velocidade do programa; alterar a sua seqüência de operações, eliminar palavras consideradas supérfluas a fim de aumentar a velocidade do processamento. Boa sorte!

16

AS PALAVRAS TURTLEGRÁFICAS

16.0 O modo TURTLE

Faz parte do sistema GraFORTH, um utilitário com o nome TURTLE, o qual consta de alguns novos comandos baseados na linguagem LOGO. Esses comandos oferecem uma maneira diferente de especificar como desenhar uma figura. O conceito é o seguinte: imagine que você está sentado num penhasco à beira de uma praia de areia branca, sem poluição. Você observa uma tartaruga se movendo na areia lisa desta praia, deixando um rastro que representa as mais confusas figuras. Vamos supor agora que você pudesse controlar telepaticamente os movimentos da tartaruga, induzindo-a a caminhar tantos passos para frente ou para trás, virar tantos graus à direita ou à esquerda, e até saltar. Com esse raciocínio em mente, você poderá obter infinitos gráficos e desenhos na areia da praia. Note que, nesta analogia, a praia corresponde à tela do micro, a tartaruga um cursor imaginário e os rastros os segmentos traçados.

Matematicamente falando, este modo de utilização de gráficos nada mais é do que uma mudança do sistema cartesiano, que vínhamos usando, para um sistema de coordenadas polares relativas.

Para inicializar o modo TURTLE, devemos compilar o texto em disco e depois chamar a palavra TURTLE que estabelece uma janela de texto com as últimas quatro linhas, reinicializa o modo gráfico, limpa a tela, estabelece cor número 3 (branca) e posiciona a "tartaruga" no centro da tela:

```
Ready READ "TURTLE "
Ready TURTLE
```

16.1 PENUP

Vamos aprender a comandar essa tartaruga.

Existem dois modos nos quais a tartaruga pode ser movimentada: movimentando deixando rastro ou movimentando sem deixar rastro.

O comando PENUP estabelece que a "caneta deve ser levantada", ou seja, a tartaruga se move sem deixar rastro na tela.

16.2 PENDOWN

O comando PENDOWN estabelece que a "caneta deve ser baixada", ou seja, sempre que a tartaruga seja movimentada, ela deixa rastro, a linha da trajetória na tela, até que o comando PENUP seja estabelecido. O modo PENDOWN é o modo estabelecido quando inicializamos com a palavra TURTLE.

16.3 MOVE

Vamos fazer essa tartaruga trabalhar. Podemos começar com ela se movimentando 50 passos (pontos) com a palavra MOVE.

```
Ready 50 MOVE
```

Na inicialização pela palavra TURTLE, a tartaruga está direcionada para cima. Quando você entrou com a instrução acima, na verdade, ordenou a ela que andasse 50 pontos em frente na direção para a qual estava apontada. Resumindo, a palavra MOVE espera na pilha quantos passos a tartaruga deve se mover na direção atual.

16.4 TURN

Deste modo, se você não possuir um meio de mudar a direção da nossa tartaruga, ela só desenhará uma reta vertical. É através da palavra TURN que você especifica para qual direção ela deve desviar. TURN espera, na pilha, o valor do ângulo (em graus) que deve ser acrescentado ao ângulo atual no sentido horário.

Para exemplificar, vamos desenhar um triângulo equilátero. Para isso, entramos com as seguintes instruções:

```
Ready TURTLE
Ready 50  MOVE
Ready 120 TURN 50 MOVE
Ready 120 TURN 50 MOVE
```

Vejamos como programar usando essas palavras e os efeitos interessantes que se podem obter com as palavras do TURTLE. Defina a palavra TRIANGULO que faz o que acabamos de executar:

```
: TRIANGULO
  50 MOVE  120 TURN
  50 MOVE  120 TURN
  50 MOVE  120 TURN ;
```

Entre com o seguinte programa, execute e verifique o seu efeito na tela:

```
: EFEITO-TURTLE
  12  0  DO
        TRIANGULO
        30  TURN
  LOOP ;
```

Com este outro programa, você consegue várias espirais na tela, bastando que você introduza na pilha 3 valores, respectivamente, para incremento, ângulo e tamanho inicial do lado:

```
: ESPIRAL
  TURTLE
  BEGIN
          DUP  MOVE  SWAP
          DUP TURN
          SWAP  DUP  4  PICK  +
  SWAP  50  >
  UNTIL
  DROP  DROP  DROP  ;
```

Para testar a palavra ESPIRAL, entre com o seguinte exemplo:

```
Ready  5  45  5  ESPIRAL
```

Esta outra rotina, a seguir, desenha uma seqüência de cinco estrelas:

```
: STAR
  TURTLE
  PENUP
  100  100  MOVETO
  PENDOWN
  5  0  DO
        8  0  DO
             50  MOVE
             144  TURN
        LOOP
        -144 TURN
  LOOP ;
```

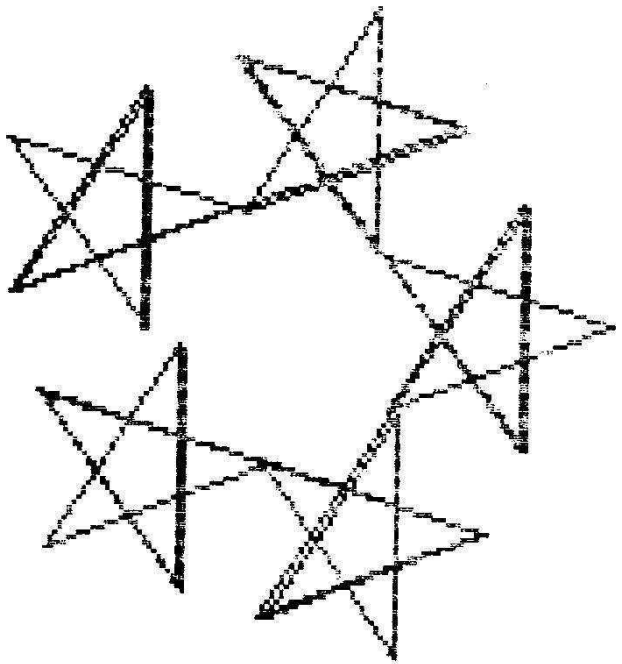



Figura 16.0 – Estrelas com comandos TURTLE.

16.5 MOVETO e TURNTO

Nós, até agora, usamos as palavras relativas do TURTLE, isto é, as palavras de tratamento de ângulos de giros e de passos de movimento incrementados a partir de uma posição anterior. Porém, o subsistema TURTLE do GraFORTH nos permite um endereçamento absoluto. A palavra MOVETO, por exemplo, permite que movamos a tartaruga desde a atual posição até uma final especificada pelas coordenadas x e y. Por exemplo, podemos selecionar onde vamos desenhar o triângulo:

```
Ready  PENUP
Ready  30  70  MOVETO PENDOWN TRIANGULO
```

Com isso, a tartaruga se movimenta até a posição (30,70) sem deixar rastro (PENUP) e se prepara para desenhar o triângulo a partir desta posição, pela palavra TRIANGULO.

Também podemos especificar que a tartaruga gire para uma posição específica. Os ângulos, dados em graus e incrementados no sentido horário, têm seu valor zero direcionado para cima, 90 corresponde à direita, 180 para baixo e 270 para a esquerda. Para isso usamos a palavra TURNTO.

Estas duas palavras são geralmente utilizadas no posicionamento de uma figura em qualquer parte da tela. Vejamos exemplos de utilização desta e de outras palavras na montagem de um cenário. Inicialmente, montemos uma casa:

```
: CASA
   45 TURN 20 MOVE
  -45 TURN 20 MOVE
  -45 TURN 15 MOVE
  -90 TURN 35 MOVE
  -45 TURN 20 MOVE
  -90 TURN 20 MOVE
  -90 TURN 20 MOVE
   45 TURN 20 MOVE PENUP
  180 TURN 20 MOVE PENDOWN
   90 TURN 15 MOVE ;
```

Podemos também montar uma igreja, adicionando à casa uma cruz no seu teto:

```
: IGREJA
  CASA
  45 TURN 15 MOVE PENUP
  150 TURN  7 MOVE PENDOWN
  120 TURN  7 MOVE ;
```

Com duas casas e uma igreja, podemos montar uma pequena vila:

```
: VILA
  PENDOWN CASA 90 TURNTO PENUP
  40 MOVE  0 TURNTO
  PENDOWN CASA 90 TURNTO PENUP
  40 MOVE  0 TURNTO
  PENDOWN IGREJA ;
```

Agora podemos montar uma cidade:

```
: CIDADE
  PENUP  0 TURNTO 40 120 MOVETO VILA
  PENUP  0 TURNTO 100 120 MOVETO VILA
  PENUP  0 TURNTO 160 120 MOVETO VILA ;
```

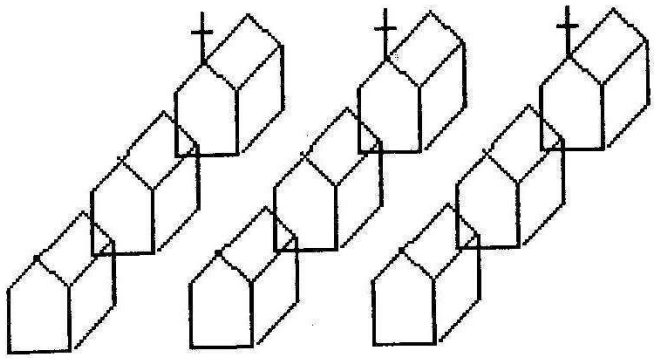



Figura 16.1 -- Cidade produzida com comandos TURTLE.

Note a utilização de MOVETO e TURNTO para que a palavra VILA possa desenhar vila em qualquer posição da tela, inclusive com outra inclinação. Tente:

```
Ready PENUP 45 TURNTO 60 60 MOVETO VILA
```

16.6 Conclusão

Finalizando nossa abordagem sobre o utilitário TURTLE, vamos apenas dizer algo mais a respeito da linguagem LOGO, que utiliza de maneira substancial a geometria da tartaruga, chegando inclusive a se confundirem numa só.

O LOGO foi idealizado pelo professor Seymour Papert, no Instituto Tecnológico de Massachusetts (MIT), na década de 70, com o intuito de servir como ferramenta de ensino utilizando computadores. A idéia era proporcionar total liberdade de criação ao aluno, ao contrário dos moldes tradicionais de ensino. A linguagem teria de ser muito fácil de aprender e ser suficientemente poderosa a fim de permitir que o aluno explorasse todas as atividades que se podem fazer com um computador.

Sob este ponto de vista, podemos acrescentar que a versão dos comandos da geometria da tartaruga, incluídos aqui, e toda a versatilidade natural do FORTH permitem ao GraFORTH a sua utilização em substituição ao LOGO no ensino por computador, no que diz respeito à potencialidade de recursos e liberdade de criação.

Parte sexta

Os caracteres gráficos

17

INTRODUÇÃO AOS CARACTERES GRÁFICOS

17.0 Apresentando os caracteres gráficos

No Capítulo 14, vimos como são montados os caracteres para serem exibidos na tela, a partir de uma página de alta resolução (caso do PLOT-A). Nosso objetivo agora é mostrar algumas características e vantagens do sistema GraFORTH na manipulação de caracteres e sugerir algumas possibilidades de aplicação de caracteres especiais na animação de desenhos.

Inicialmente, vamos apresentar os caracteres de controle, aqueles fornecidos pelas teclas utilizadas conjuntamente com a tecla ⟨CTRL⟩ (ou ⟨CONTROL⟩), de uso especial pelo sistema GraFORTH. Existem caracteres que só devem ser usados de dentro de um programa, e não entrados diretamente através do teclado (neste caso, o GraFORTH tentará lê-los como caracteres dentro de uma palavra do sistema). Outros caracteres não imprimíveis podem ser usados dentro de uma definição de palavra quando você quiser protegê-la, ou seja, não podendo ver num LIST todos os caracteres do nome da palavra que só você conhece (veja o Capítulo 7).

17.1 As fontes do GraFORTH

Os caracteres, todos juntos, formam um conjunto que é conhecido como ALFABETO ou FONTE. As fontes diferem entre si no formato de cada caractere e nos caracteres que contêm, formando um dado estilo de escrita. Você já deve ter notado que quando estabelece o modo TEXT, o prompt Ready aparece em maiúsculas, ao invés de em letras minúsculas como normalmente. Isto se deve ao fato de o Apple II (dependendo do sistema Apple II que você tem) possuir somente letras maiúsculas no seu hardware, sendo estas utilizadas pelo GraFORTH nesse modo. Todavia, no modo gráfico, o sistema GraFORTH utiliza um conjunto de caracteres da memória que está armazenado em arquivo binário no disquete do sistema e que é lido para a memória quando o sistema é inicializado.

Você poderá, ao longo deste capítulo, criar seu próprio conjunto e estilo de caracteres. Enquanto isso, utilize os caracteres disponíveis no disquete do sistema:

Quadro 17.1 **Fontes do sistema GraFORTH**

CHR.SYS
CHR.STOP
CHR.SLANT
CHR.GOTHIC
CHR.BYTE
CHR.STUFF
CHR.MAXWELL

Figura 17.1 — Todos os conjuntos de caracteres.

Esses conjuntos de caracteres são fartamente apresentados nos demonstrativos, sendo que os dois últimos apresentam caracteres usados em animação, que mais tarde você poderá manipulá-los bem.

Uma vez na memória, o conjunto de caracteres ocupa 768 bytes: são 96 caracteres imprimíveis, cada um deles usando 8 bytes, que nada mais são do que um "quadro" gráfico de caractere, como descrito no item 14.3 do Capítulo 14. Quando o sistema GraFORTH é inicializado, ele carrega a fonte CHR.SYS para a memória, iniciando no endereço 2048. Sempre que o sistema GraFORTH for exibir um caractere no vídeo, ele procura nesse endereço o "quadro" que representa o caractere e coloca esses bytes na região de memória correspondente da página de alta resolução, resultando na apresentação do caractere na tela.

17.2 Selecionando as fontes

Mais de um conjunto de caracteres podem estar presentes ao mesmo tempo na memória. Basta que carreguemos do disquete para uma região livre da memória. A região de memória comumente utilizada para este fim tem início em 2816 e tem um comprimento de 3072 bytes (exatamente 3 Kbytes).

Para carregar o conjunto de caracteres, basta utilizar o comando do DOS:

```
Ready  CR  132  PUTC PRINT " BLOAD CHR.BYTE,A2816 " CR
```

ou utilizar o programa DOS listado no item 13.6 do Capítulo 13.

Após cessar a operação do drive, o conjunto de caracteres é armazenado a partir do endereço 2816. Para poder utilizar o estilo "BYTE", devemos alterar o endereço do conjunto de caracteres através da palavra CHRADR (contração de CHaRacter ADdRess) para o endereço onde armazenamos essa fonte:

```
Ready  2816  CHRADR
```

A partir de agora, os novos caracteres exibidos serão no estilo "BYTE", porém os que já haviam sido mostrados permanecem inalterados. No entanto, se for executado um scroll de tela, todos os caracteres serão reexibidos uma linha acima no mais novo conjunto especificado.

Os caracteres ocupam uma faixa dentro dos códigos ASCII que vai do número 160 ao 255. Desta forma, podemos exibir todos os caracteres através da palavra PUTC:

```
Ready  256  160  DO  I  PUTC  LOOP
```

Já foi dito que você pode trabalhar com várias fontes de caracteres ao mesmo tempo, bastando que as carregue em outras áreas livres da memória. Seu uso pode ser facilmente estabelecido e trocado através da palavra CHRADR. É bom lembrar-se sempre que cada conjunto ocupa 768 bytes e só existem 3072 bytes livres, ou seja, somente 4 conjuntos de caracteres (fora o normal do sistema) podem ser utilizados simultaneamente, sob pena do conjunto sobrepor-se ao sistema e você perder todo o controle.

Para voltar ao conjunto normal do sistema (CHR.SYS), basta entrar:

```
Ready  2048  CHRADR
```

ou então utilizar uma palavra já contida no GraFORTH (CHRSET) que simplesmente devolve à pilha o valor 2048, correspondendo ao endereço do conjunto normal.

```
Ready  CHRSET  CHRADR
```

Se você preferir, é possível também carregar outra fonte neste endereço, que passará a ser usado como fonte normal para o sistema. Assim, para exemplificar, passaremos a usar o conjunto CHR GOTHIC como a fonte normal:

```
Ready CR 132 PUTC PRINT " BLOAD CHR.GOTHIC,A2048 " CR
```

Agora, só será restabelecida a antiga fonte se ela for recarregada do disquete.

```
Ready  CHRSET  CHRADR
```

Estabelecido o endereço normal, observaremos e utilizaremos o estilo gótico, a menos que você não consiga se acostumar com essas letras rebuscadas. Se assim for, veja se já consegue restabelecer o conjunto de caracteres CHR.SYS, seguindo o que já foi explicado.

18

CHAREDITOR: EDITOR DE CARACTERES

18.0 Editando caracteres com CHAREDITOR

Conforme já mencionamos, o sistema GraFORTH possui a habilidade de tratar figuras como caracteres. Possuindo vários estilos de letras, você mesmo pode descrever suas próprias letras, usando um estilo próprio. Neste capítulo, pretendemos mostrar-lhe como criar seu próprio alfabeto personalizado.

O CHAREDITOR é um arquivo de texto gravado no disquete do sistema que uma vez compilado proporciona um ambiente propício para desenvolvimento e criação de caracteres. Ele é um programa grande e é aconselhável esquecer (FORGET) palavras que você tenha definido anteriormente. Para isso, execute a palavra LIST e procure a palavra CHS (a última definida no sistema original) e de um FORGET para a palavra listada acima dela.

Para carregar CHAREDITOR do disquete do sistema, entre:

```
Ready  READ " CHAREDITOR "
Ready  HOME  RUN
```

Note que usamos a palavra HOME para limpar a tela antes de executar o programa. Isso se faz necessário porque o CHAREDITOR não o faz automaticamente. Observe a vantagem que se obtém com esse fato. Figuras criadas por outros programas podem ser retidas e usadas dentro do CHAREDITOR, estimulando-o a utilizá-las como caracteres nos seus programas.

O programa, inicialmente, apresenta-nos um menu de comandos à direita do vídeo, uma mensagem avisando ao usuário que ele está a espera de um comando e um cursor (ponto) piscante no canto superior esquerdo, que será usado para a criação e edição de seus desenhos e formas de caracteres.

A seguir, apresentaremos a função de cada um dos comandos, as mensagens apresentadas e as modificações ocorridas.

Os primeiros comandos apresentados no menu dizem respeito à manipulação do cursor, ou seja, a sua movimentação dentro do bloco

```
I=Up
J=Left
K=Right
M=Down
P=Plot
U=Unplot
L=Line
Z=Zline
C=Color
A=Address
T=Transfer
B=Blocksize
E=Erase
R=Read
W=Write
S=Save
G=Get
D=Disp. Chars
X=Step Size
Q=Quit

Enter command
```

Figura 18.0 – Tela principal do CHAREDITOR.

definido. É fácil observar que o cursor anda somente dentro dos limites do bloco especificado. Na inicialização do CHAREDITOR, o tamanho especificado é de um caractere.

18.1 Comandos I=Up, J=Left, K=Right, M=Down

Os comandos são os mesmos usados no Applesoft BASIC juntamente com a tecla escape: I, J, K, M (sendo que não se deve usar a tecla escape, pois o CHAREDITOR só entende I, J, K e M). Os sentidos são os especificados na figura 18.1 e são usados para a movimentação do cursor.

18.2 Comando X=Stepsize

Inicialmente, o cursor só se movimenta de um em um ponto. Porém, este passo pode ser aumentado quando se desejar um deslocamento maior ou mais rápido. Para isso, utilizamos o comando X (penúltimo na lista do menu) que nos apresenta a seguinte mensagem:

Enter step size:

onde se espera que você digite um número e a tecla ⟨RETURN⟩, especificando de quantos em quantos pontos o cursor passa a se deslocar. Se por acaso você der um valor acima do tamanho do bloco, o cursor se movimentará somente até a borda do bloco.

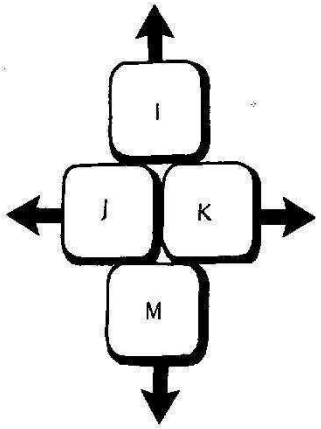


Figura 18.1 – As teclas I, J, K e M no CHAREDITOR.

Você então pode observar o espaço que possui para se movimentar, andando de um canto para outro do bloco, especificando um valor alto, 250 por exemplo, e utilizando as teclas I, J, K e M para se movimentar.

18.3 Comando P=Plot

O comando P (PLOT) é usado para plotar o ponto na posição atual do cursor. Restabeleça o valor 1 para o passo do cursor através do comando X e comece a plotar pontos dentro desse caractere. Tente, por exemplo, desenhar a letra "A" como foi feito no item 14.3 do Capítulo 14.

18.4 Comando U=Unplot

Cometeu um erro? Bem, para facilitar a correção de erros, melhoramentos de arte final, sem ter de apagar tudo e recomeçar, existe o comando U, "UNPLOT" (trabalhando da mesma forma como o seu homônimo que foi visto quando falamos de palavras gráficas bidimensionais), que apaga o ponto atualmente posicionado pelo cursor.

18.5 Comando L=Line

Há ocasiões em que temos a necessidade de plotar vários pontos consecutivos numa dada direção. Para facilitar nosso trabalho, o

comando L traça uma linha, desde a última posição especificada por um comando P, U ou outro L até a posição atual do cursor (note que é evidente a semelhança com a forma como são usadas as palavras gráficas bidimensionais).

18.6 Comando Z=Zline

É o comando que é análogo a UNLINE, ou seja, apaga uma linha traçada, desde o último ponto especificado por P, U, L ou outro Z até a posição atual do cursor.

Agora que você já conhece estes comandos, tente definir a letra "Z" minúsculo, por exemplo.

18.7 Comando C=Color

Com este comando, você pode selecionar a cor ou as cores do caractere através da mensagem:

```
Enter color (0-7) :
```

que espera um número com a mesma correspondência com os códigos de cores já mostrados no Quadro 15.1 do Capítulo 15.

18.8 Comando A=Address

Comando usado para especificar o endereço inicial do conjunto de caracteres a serem utilizados. É útil quando há mais de um conjunto de caracteres na memória. O endereço-padrão – 2816 – aparece na mensagem apresentada, sendo desnecessário entrar com outro valor se ele for mantido:

```
Enter charset work area address : 2816
```

Neste ponto, vamos abrir um parênteses para esclarecer algumas idéias sobre as possibilidades de manipular e alterar os endereços de um conjunto de caracteres. Uma idéia interessante é alterar, aumentando ou diminuindo, em múltiplos de oito, o endereço inicial do conjunto de caracteres para obtermos o estranho efeito de, por exemplo, apertarmos a tecla correspondente à letra "A" e aparecer, no vídeo, a letra "B". Ou ainda, se essa alteração não for em múltiplos de oito, podemos colocar um caractere com uma parte de uma letra e outra parte de uma outra.

Porém, quando estamos construindo nosso próprio conjunto de caracteres, não nos é interessante que esses "efeitos" ocorram. Para evitar isso, devemos saber a seqüência certa dos caracteres ASCII. Assim, mostramos, na figura 18.2, os caracteres do padrão ASCII e como deve aparecer sua posição relativa num conjunto de caracteres especificados pelo GraFORTH:

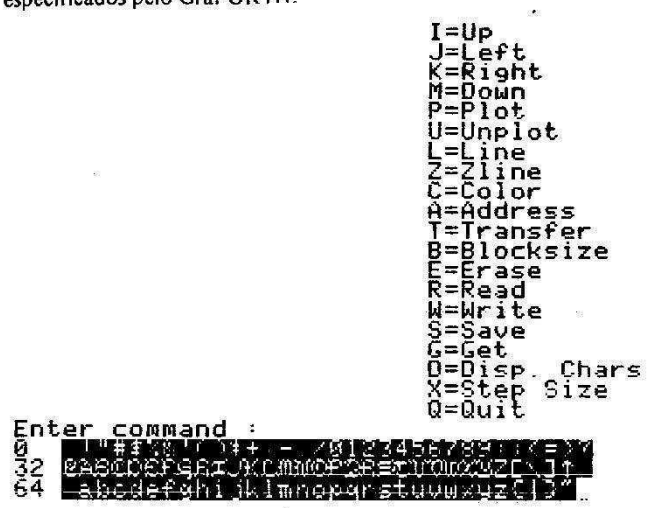


Figura 18.2 – Tela principal de CHAREDITOR com conjunto de caracteres.

18.9 Comando T=Transfer

Comando utilizado para transferir blocos ou conjuntos de caracteres de uma região para outra dentro da memória disponível (cuidado para não mover para uma área dentro do sistema).

```
charset Transfer
enter "FROM" address :
enter "TO" address :
enter Length (normally 768) :
```

As mensagens, que se sucedem, pedem os dados necessários à transferência dos bytes. Inicialmente, é pedido o endereço inicial do conjunto a ser transferido. Depois, o endereço para onde vai ser levado e, por último, o comprimento do bloco (normalmente, 768 bytes se for transferido o conjunto inteiro).

Este comando muito nos auxilia quando temos figuras similares, com pequenas mudanças, onde não é preciso sua redefinição total, mas simplesmente alterar a parte correspondente.

18.10 Comando B=Block Size

O comando B é o que estabelece o subconjunto do conjunto de caracteres — o bloco de caracteres — que vamos utilizar para construir uma figura. Na verdade, ele especifica uma região retangular cujos comprimentos são medidos em números de caracteres. Assim, se aparecerem as mensagens:

Enter block horizontal size :
Enter block vertical size :

responderemos respectivamente 3 e 2, pois nosso bloco terá um tamanho de três colunas de caracteres por duas linhas. Podemos visualizar esta área delimitada por quatro pontos nos cantos do bloco, que estabelecem melhor visualização do espaço de manipulação do cursor, sendo que esses pontos são externos ao bloco.

18.11 Comando E=Erase

Comando utilizado quando desejamos apagar o bloco para reinicializar um desenho que apresenta algum erro ou que não satisfaz ao que era esperado ou desejado. Como é um comando que pode apresentar conseqüências desastrosas caso seja entrado por acaso, ele apresenta uma mensagem pedindo uma confirmação da decisão tomada:

Erase (Y/N):

Se realmente desejar que o bloco mostrado na tela seja apagado, você responderá teclando Y, caso contrário, N anulará o comando.

18.12 Comando R=Read

É o comando que apresenta, na área de movimentação do cursor, os caracteres já definidos no conjunto especificado que faz parte do bloco no qual você está trabalhando. Isto permite alteração, visualização, ou verificação do conteúdo do bloco.

Para especificar que caracteres do conjunto vão formar o bloco,

você deve entrar com o número (seguindo a numeração apresentada na descrição do comando A) do primeiro caractere do bloco, ao responder a mensagem:

Enter character number to read :

Todos os caracteres seguintes a este, necessários para formar o bloco atualmente definido pelo comando B, serão mostrados na área de trabalho.

18.13 Comando W=Write

Uma vez pronto, montado e editado, o seu bloco deve ser passado da área de trabalho para a memória, onde está localizado o conjunto de caracteres atualmente definido pelo comando A. Esta é a função do comando W e funcionalmente corresponde ao inverso do comando R, necessitando, inclusive, do número da posição do caractere inicial, que deve ser introduzido após a mensagem:

Enter character number to be written :

18.14 Comando S=Save

Salva, num arquivo em disco, o conjunto de caracteres atualmente especificado pelo comando A (de comprimento de 768 bytes) com nome especificado através da mensagem:

Enter save file name :

Este arquivo, assim criado, pode ser acessado normalmente pelos comandos do DOS, lembrando-se unicamente que é um arquivo binário e não de textos.

18.15 Comando G=Get

É o negativo do comando S, ou seja, você especifica através da seguinte mensagem:

Enter Load file name :

o nome do arquivo binário gravado em disco que será carregado na memória a partir do endereço estabelecido pelo comando A com o comprimento correspondente de 768 bytes.

18.16 Comando D=Display

Comando utilizado para se obter uma visão geral dos caracteres que estão sendo usados. Mostra em 3 linhas na parte inferior do vídeo, numeradas em 0, 32, 64 (sendo, portanto, 32 caracteres por linha), em vídeo reverso, todos os caracteres definidos atualmente. Muito útil para se ter uma visão geral e também para localizar o número correspondente a determinado caractere.

18.17 Comando Q=Quit

Abandona o CHAREDITOR, retornando ao sistema GraFORTH com a mensagem inicial de autoria do sistema.

19

MANIPULAÇÃO DE BLOCOS DE CARACTERES

19.0 Manipulando os caracteres gráficos

Você pode manipular blocos, ou conjuntos, de caracteres de dentro do sistema normal do GraFORTH da mesma forma que manipulava de dentro do CHAREDITOR (na verdade, é o CHAREDITOR que utiliza estas palavras que já estão definidas no sistema GraFORTH para essa manipulação). Para isto, basta você ter armazenado, numa área de memória, o conjunto de caracteres gráficos anteriormente criados pelo CHAREDITOR.

Devemos também ter em mente que esses caracteres (se estamos querendo trabalhar com desenhos e não com letras) não servem para a escrita – principalmente para a visualização dos comandos introduzidos – se só se encontram figuras neste conjunto. Assim, é boa prática manter um grupo de letras (por exemplo, todas as letras maiúsculas, dígitos e símbolos mais usados) junto com os desenhos ou então alternar os endereços de dois conjuntos de caracteres conforme se queiram desenhos ou palavras a serem exibidos na tela.

19.1 Utilizando caracteres em animação gráfica

O sistema GraFORTH, ao apresentar a possibilidade de manipular blocos de caracteres, adquire a função de um potente editor de animação, como vários outros existentes no mercado para a linha Apple II. Tudo, na verdade, baseia-se no esquema tradicional, usado na televisão, no cinema e em desenhos animados: uma sucessão, em pequenos intervalos de tempo, de fotogramas que impressionam a retina, dando a sensação de movimento.

Dos "animadores" comerciais existentes, encontramos três maneiras de se fazer a seqüência de animação: tabela de forma, movimento de blocos e movimentos de transparências.

A primeira não é utilizada pelo GraFORTH por apresentar graves inconvenientes: é demasiadamente lenta, pois a forma deve ser desenhada a cada nova posição em que ela se apresente, além de problemas ao ser sobreposta a um fundo colorido.

As transparências (sprites) também não são manipuladas pelo GraFORTH. Sua implementação é complexa e demandaria, talvez, um sistema único (não uma linguagem gráfica como o GraFORTH) para manipulá-las eficientemente. A vantagem nítida das transparências (como o próprio nome indica) é que só é sobreposto o desenho contido no bloco de memória, sendo as partes do bloco externas ao contorno do desenho "transparentes".

Desta forma, é através dos comandos de manipulação de blocos que podemos construir animações em duas dimensões com o GraFORTH, da mesma maneira que qualquer outro aplicativo existente, e, porque não dizer, com muito mais recursos disponíveis.

19.2 Cor e tamanho dos caracteres

Antes de tratarmos com um bloco de caracteres propriamente, temos de ver alguns atributos de que o sistema GraFORTH dispõe ao tratar cada caractere individualmente, pelo fato de eles serem "desenhados" na tela no modo gráfico de alta resolução, ao invés de tela no modo texto. Deve ficar clara a posição de que essas propriedades seguintes só são possíveis devido a esse fato.

Os caracteres, através do GraFORTH, podem ser apresentados em vários tamanhos diferentes, sendo todos eles múltiplos de um caractere. Estes tamanhos são selecionados através da palavra CHRSIZE, que espera, na pilha, o tamanho com o qual se deseje escrever as letras. Os tamanhos válidos estão entre 0 e 8.

tamanho CHRSIZE

O tamanho zero corresponde à saída de caracteres normais, sem coloração. Os valores seguintes correspondem ao número de caracteres normais que cada caractere passa a ocupar. Por exemplo, 3 CHRSIZE estabelece um caractere ocupando 3 por 3 caracteres normais. O tamanho 1 é o mesmo que o tamanho 0 (1x1 caractere), porém utiliza as capacidades gráficas do GraFORTH, podendo, assim, ser coloridas.

Pode-se observar, utilizando os caracteres com tamanhos ampliados, que os tempos de impressão e de scrool da tela são mais demorados.

Outro fato que deve ser observado é o de que, com o aumento do tamanho do caractere, menos caracteres podem ser exibidos na tela. Para evitar problemas, janelas de texto com as dimensões permitidas são automaticamente selecionadas quando do uso da palavra CHRSIZE. A seguir, uma tabela apresentando os números dos caracteres possíveis nas janelas de texto:

TAMANHO	COLUNAS	LINHAS
0	40	24
1	32	24
2	16	12
3	10	8
4	8	6
5	6	4
6	5	4
7	4	3
8	4	3

Outra capacidade no tratamento de caracteres, que interessa basicamente a quem possui um monitor de vídeo colorido ou uma televisão colorida, é claramente a facilidade de se escolher e alterar a cor de cada caractere exibido. O esquema é o mesmo usado no tratamento de cor das figuras bidimensionais tratadas no capítulo anterior, e as considerações no que se diz respeito à mistura de cores e de grupos de cores devem ser analisadas e respeitadas.

Se você possui um monitor colorido, vamos exemplificar o que pode ocorrer ao se manipular texto alterando-se a sua cor. Inicialmente, vamos apagar a tela toda e, se sua janela de texto não ocupa a tela toda, é bom estabelecê-la deste modo com:

```
Ready  0  40  0  23  WINDOW
```

e estabelecer a cor verde (desde que anteriormente CHRSIZE estabeleceu um tamanho de 1 a 8):

```
Ready  HOME  1  COLOR
```

Feito isso, o prompt Ready aparecerá com a cor verde. Tente agora alterar a cor verde para violeta:

```
Ready  2  COLOR
```

Agora tecle ⟨RETURN⟩ algumas vezes para obter um scrool do texto na tela. Com isso, você pode notar que o prompt escrito anteriormente com verde não responde ao scrool, pois somente caracteres com a mesma cor e o mesmo tamanho respondem aos comandos de texto estabelecidos.

Efeitos interessantes podem ser obtidos, combinando as opções de cores e tamanhos dos caracteres e alterando-os com a palavra INVERSE.

Por exemplo:

```
Ready  HOME 5 CHRSIZE 5 COLOR INVERSE
```

Esta instrução estabelece a limpeza da tela, tamanho 5 e cor 5 para os caracteres, no modo inverso. Veja o resultado com o prompt e o cursor piscando que aparecem na primeira linha da tela.

Porém, este é um modo extremamente desagradável de se introduzir comandos ou definir novas palavras, sendo que sua utilização é realmente recomendada para ilustrar gráficos, desenhos e jogos. Para voltar à situação normal, entre:

```
Ready  NORMAL 3 COLOR 0 CHRSIZE ERASE
```

ou de uma forma rápida, reinicialize todo o sistema, pressionando as teclas ⟨CTRL⟩⟨RESET⟩ (ou somente a tecla ⟨RESET⟩, dependendo do seu sistema Apple II) ou entrando:

```
Ready  ABORT
```

A palavra ABORT gera o mesmo efeito, executado somente após um scrool da tela.

19.3 Especificando o bloco

Uma vez que, através do CHAREDITOR, definimos nossos blocos de caracteres com as figuras que criamos, devemos definir o tamanho desse bloco, da mesma forma como de dentro do CHAREDITOR, especificando largura e comprimento em termos de números de caracteres.

Para exemplificar, vamos carregar uma das fontes gravadas no disquete do sistema que possui caracteres gráficos, entrando:

```
Ready  CR 132 PUTC PRINT " BLOAD CHR.STUFF,A2816 " CR
```

ou usando o programa DOS definido no item 13.6 do Capítulo 13.

Poderemos, com esse conjunto de caracteres, exibir e simular movimentos de faces e helicópteros. Inicialmente, vamos estabelecer o tamanho de uma face:

```
Ready  3 2 BLKSIZE
```

A sintaxe usada foi:

```
num1  num2  BLKSIZE
```


onde o num1 é o número de caracteres horizontais e o num2 é o número de caracteres verticais. Ou seja, um bloco de num1 caracteres por linha com num2 linhas. É importante, sempre que for utilizar um novo bloco de tamanho diferente, alterar os parâmetros de BLKSIZE. Lembre-se também que a palavra ABORT não restaura BLKSIZE.

19.4 Exibindo o bloco

Para exibir um bloco de caracteres na tela, você necessita ter os caracteres na memória, estabelecer o endereço desses caracteres, estabelecer o tamanho que o bloco vai ter e quais caracteres pertencem ao bloco que vai ser exibido. As três primeiras operações já sabemos fazer. A última é simples. Só nos basta determinar o primeiro na seqüência de caracteres contida na fonte que corresponde ao canto superior esquerdo do bloco. Os caracteres seguintes em número igual ao número de caracteres no bloco também fazem parte do bloco na seqüência completando uma linha, partindo para a linha seguinte e assim sucessivamente. No nosso exemplo, são 6 caracteres (3x2). A palavra PUTBLK executa a operação de exibir um caractere com o tamanho definido por BLKSIZE na posição corrente do cursor (correspondendo ao primeiro caractere, canto superior esquerdo) através das funções normais de posicionamento HTAB e VTAB. A palavra PUTBLK espera, na pilha, o número de início dos caracteres a serem exibidos, da mesma forma como era especificado no CHAREDITOR. Exibiremos uma face:

```
Ready  3  2  BLKSIZE
Ready  HOME
       2816 CHRADR 78 PUTBLK
       CHRSET CHRADR 12 VTAB
```

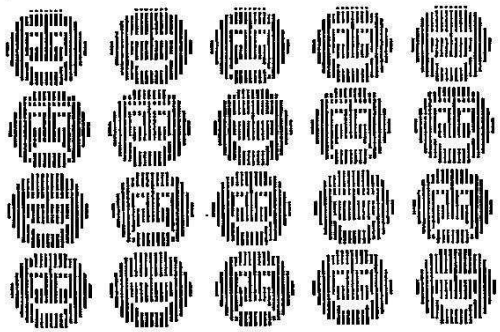


Figura 19.1 — As faces — bloco de caracteres.

Vejamos o resultado dessa seqüência de comandos. A instrução acima limpa a janela de textos, endereça o conjunto de caracteres que possui as palavras gráficas, exibe um bloco de 3x2 caracteres, com caracteres iniciando na posição 78, restabelece o conjunto normal e posiciona para a linha 12 da janela de texto, exibindo o prompt "Ready" para que ele não se sobreponha à imagem da face que se apresenta no canto superior esquerdo, uma vez que PUTBLK não altera a posição do cursor.

Ressaltemos mais uma vez a importância de restabelecer os endereços para o conjunto de caracteres que estamos usando no momento anterior e posterior da impressão, para não termos o desprazer de teclar uma letra e obter um olho! Porém, devemos facilitar o nosso trabalho para não ter que digitar todas as vezes esses endereçamentos, através das seguintes palavras:

```
Ready : INICIO HOME 2816 CHRADR ;
Ready : FIM CHRSET CHRADR 12 VTAB ;
```

Entre uma e outra palavra, podemos estabelecer quaisquer instruções de impressão de blocos desde que sejam todos impressos na mesma posição: canto superior esquerdo.

Dissemos anteriormente que o princípio da animação está na superposição de imagens. Então, através de um loop simples de impressão de alguns desenhos no mesmo lugar, podemos criar uma pequena animação. Veja:

```
Ready 5 3 BLKSIZE
Ready INICIO 100 0 DO
          33 PUTBLK 48 PUTBLK 63 PUTBLK
      LOOP FIM
```

Analise como você pressente o movimento das hélices do helicóptero, através das impressões de três blocos de caracteres representando três posições diferentes das hélices do helicóptero e devido à velocidade com a qual PUTBLK é executado.

Assim como você deseja a noção de movimentação pela tela — não impressões consecutivas na mesma posição — precisa também ir incrementando as posições com VTAB e HTAB e imprimindo os blocos correspondentes. Porém, se você não apagar os blocos nas posições anteriores, ocorrerá que será deixado um rastro atrás de você. Experimente o seguinte exemplo, que vai desenhando o helicóptero se movimentando numa linha da tela, porém, deixando, atrás de si, o rastro indesejável de helicópteros:

```
Ready 5 3 BLKSIZE
```



```
Ready INICIO 35 0 DO
          I HTAB 33 PUTBLK 48 PUTBLK
          63 PUTBLK
      LOOP FIM
```

Para evitar esse problema, basta que, antes de desenhar uma nova posição, apaguemos o bloco anterior. Isto é feito através da palavra UNBLK, que não necessita de nenhum valor na pilha nem de nenhum conjunto de caractere específico, pois simplesmente coloca espaços em branco na área delimitada por BLKSIZE.

Tentemos, novamente, usando UNBLK agora:

```
Ready INICIO 0 34 DO
          UNBLK I HTAB 33 PUTBLK
          48 PUTBLK 63 PUTBLK
      -1 +LOOP FIM
```

Você também poderia pensar em usar EXMODE e reimprimir o bloco, o que causaria o desaparecimento do mesmo. Porém, isso só ocorrerá se você antes der um CALL na rotina de endereço -936, que apaga a página de texto do Apple II. Deixe-nos explicar o porquê. Cada caractere impresso na página de alta resolução tem o seu valor ASCII correspondente armazenado na página de texto do Apple II. Muito bem, se quisermos imprimir um caractere que já esteja naquela posição, nada é impresso na página de alta resolução — pois, ele já está lá — ganhando tempo e tornando mais rápido o processo de impressão. Deste modo, se você tentar reimprimir no modo EXMODE para apagar o caractere, não irá conseguir porque o caractere não será impresso pelo fato explicado acima. Assim sendo, sua única solução é primeiro apagar a página de texto (sem apagar a de alta resolução) para que a impressão tenha efeito desejado.

Vamos usar o mesmo exemplo com EXMODE agora:

```
Ready INICIO 0 34 DO
          ORMODE I HTAB 33 PUTBLK
          48 PUTBLK 63 PUTBLK
          -936 CALL EXMODE 63 PUTBLK
      -1 +LOOP FIM
```

Vejamos o que ocorre quando fazemos essa movimentação sobre retas desenhadas em alta resolução:

```
Ready 10 VTAB 0 60 POSN
      250 90 LINE EXMODE
```

```
Ready  INICIO  0  34  DO
          I  HTAB  33  48  63  63  48  33
          6  0  DO
              PUTBLK  LOOP
          -1  +LOOP  FIM
```

Repare como não houve necessidade de usarmos -936 CALL, uma vez que não reimprimimos o mesmo caractere consecutivamente, mas perdemos em velocidade, pois tivemos que apagar 3 vezes o bloco.

Parte sétima

Músicas e sons no sistema GraFORTH

20

APRESENTANDO OS RECURSOS SONOROS DO GRAFORTH

20.0 Seu órgão eletrônico

Quem pela primeirva vez tem contato com o sistema GraFORTH e se inicia com os programas demonstrativos, observa o uso bem aplicado da capacidade sonora do equipamento. O GraFORTH, como uma linguagem completa, tenta obter o máximo possível dos recursos da máquina e por isso deve manter funções capazes de manipular o alto-falante do sistema Apple II.

Como animador, não poderia deixar de lado a sonoplastia: do mesmo modo que os diretores de cinema contratam orquestras inteiras para criar o clima de uma cena, você deve ser capaz de conduzir o clima de suas apresentações na tela de vídeo ligado ao seu micro.

Imagine se você estivesse jogando um flipper qualquer, ou então Zaxxon, Columbia, Pole Position, sem que saísse um som sequer daquelas máquinas caça-níqueis animadas! O efeito sonoro é um item que não pode ser esquecido na elaboração de um jogo que envolva principalmente ação.

Basicamente, o sistema GraFORTH manipula notas musicais que podem ser tocadas em nove diferentes "vozes", mas não simultaneamente. Apenas duas palavras incorporadas no sistema cuidam da manipulação do som através do alto-falante do Apple: VOICE e NOTE.

Após apresentar alguns conceitos musicais introdutórios, vamos definir novas palavras que se utilizam dessas duas e que podem cobrir a maior parte das aplicações musicais de maneira simples e eficiente.

20.1 Introdução aos conceitos musicais

Assim como a linguagem escrita constitui um meio para representar a linguagem falada, na música também há um processo para representar graficamente os sons: as notas musicais. Estas, como outros sinais musicais, são grafadas sobre um grupo de cinco linhas e quatro espaços, denominados pauta ou pentagrama.

As notas são sinais que servem para representar os sons musicais e estão fisicamente associadas a uma freqüência de vibração (correspondente à altura do som).

Embora seja muito grande o número de sons utilizados na música, empregam-se somente sete nomes para designá-los correspondentes a sete notas que se repetem nas mais diversas alturas (as chamadas "oitavas"). Na ordem ascendente, isto é, do grave para o agudo, a série das sete notas é a seguinte: DÓ, RÉ, MI, FÁ, SOL, LÁ, SI.

A figura abaixo mostra onde se localizam as notas no pentagrama:

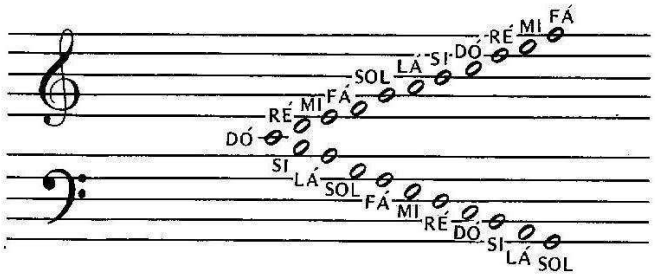


Figura 20.0 – Localização das notas no pentagrama

A diferença na freqüência de vibração entre duas notas (com exceção da diferença entre MI e FÁ e entre SI e o DÓ seguinte) é chamada TOM e corresponde a um fator de multiplicação.

No caso entre MI e FÁ e entre SI e DÓ, o fator de multiplicação é a raiz quadrada do fator anterior, e é chamado SEMITOM.

Assim, existem as notas intermediárias entre as que são separadas por um TOM. Estas são chamadas "sustenido" (#), que é um semitom acima. Como exemplo, temos DÓ# (nota entre DÓ e RÉ), RÉ#, FÁ#, SOL#, LÁ# (não existem o MI# e o SI#).

Podemos observar um teclado de piano e ver a repetição de teclas brancas e pretas (sete brancas e cinco pretas) que correspondem a oitavas. As sete teclas brancas são as sete notas e as cinco pretas são as notas intermediárias (sustenidos).

Podemos determinar o valor dos fatores correspondentes a um TOM e um SEMITOM, se soubermos que a diferença entre uma nota numa oitava e numa outra oitava acima é o dobro da freqüência (a primeira mais grave e a segunda mais aguda). Se partimos do DÓ e tocamos todas as teclas, estaremos subindo na escala musical a altura de um SEMITOM, sendo ao todo 12 semitons. Logo o fator S de multiplicação correspondente a um semitom pode ser obtido por: S^12=2. O valor de um tom corresponde a subir dois semitons, ou seja T=S^2. A solução dessas equações resulta nos fatores S=1,059 e T=1,122.

Nas músicas em geral, há sons mais longos e mais breves, bem como momentos de silêncio, em que as vozes interrompem a emissão do som ou os instrumentos cessam momentaneamente a execução. Assim, para o som ser caracterizado, além da altura é necessário especificar-se a duração. As figuras servem para representar a maior ou a menor duração do som ou do silêncio. A figura abaixo apresenta as figuras musicais atualmente usadas:

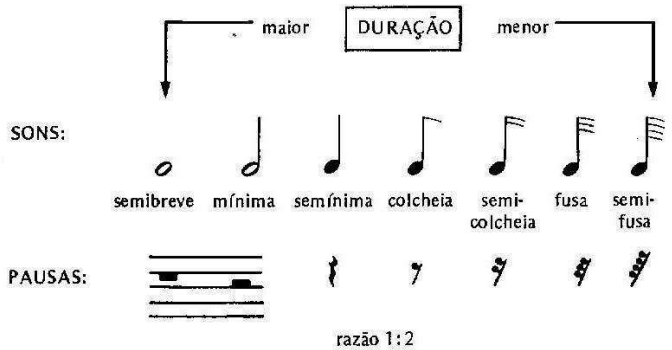


Figura 20.1 – Figuras representativas da duração dos sons e pausas

A figura com maior valor de duração é a semibreve, seguindo-se as outras, cada uma com metade do valor da anterior.

Algumas figuras podem ter a sua duração aumentada em virtude de um ponto que aparece à sua direita – conhecido por ponto de aumento – aumentando metade do valor da sua duração.

O ponto de diminuição – ponto situado acima ou abaixo de uma figura – indica que as figuras devem ser executadas como se tivessem somente a metade do valor real, isto é, separando-se os sons, como se houvessem pausas intercaladas (vide a figura 20.2).

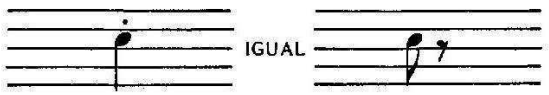


Figura 20.2 – Exemplo do efeito do ponto de diminuição

Todos esses conceitos são extremamente úteis na hora de você ter um pentagrama e tentar traduzi-lo para tocar no seu micro.

20.2 A palavra NOTE

É a palavra que deve ser usada quando desejamos que uma determinada nota seja emitida pelo alto-falante. Ao fazer isso, NOTE remove da pilha dois números – conforme mostrado abaixo – que especificam a altura e a duração para a seleção da nota desejada:

altura duração NOTE

Os números válidos para a altura e a duração estão entre 2 e 255 (apenas um byte é usado), sendo que quanto maior for o número que especifica a duração, mais longa será a nota produzida e quanto maior o número que especifica a altura, notas mais graves serão produzidas.

Abaixo vemos uma tabela apresentando as notas em cada uma das 3 oitavas possíveis com o GraFORTH e sua altura correspondente:

	3ª OITAVA	2ª OITAVA	1ª OITAVA
DO	208	104	52
DO#	197	98	49
RE	186	93	46
RE#	175	87	43
MI	165	82	41
FA	156	78	39
FA#	147	73	36
SOL	139	69	34
SOL#	131	65	32
LA	124	62	31
LA#	117	58	29
SI	110	55	27

Repare que a relação entre qualquer nota e a seguinte é aproximadamente 1,059, um semitom (exemplo: DO/DO# = 1,056, RE/RE# = 1,06).

A outra tabela a seguir mostra as durações musicais e seus valores correspondentes para serem utilizados com a palavra NOTE:

SB (semibreve)	255	SC (semicolcheia)	16
M (mínima)	128	F (fusa)	8
SM (semínima)	64	SF (semifusa)	4
C (colcheia)	32		

Assim, a semibreve possui o valor maior, sendo as outras durações seus submúltiplos.

21

MANIPULANDO OS RECURSOS SONOROS COM O GRAFORTH

21.0 Trabalhando com música

Para fazermos melhor uso da música em nossas aplicações, podemos, utilizando as tabelas do capítulo anterior, construir palavras em GraFORTH que nos dêem os valores numéricos correspondentes às alturas, oitavas e durações, sendo possível definir poucas palavras que cubram todas as possibilidades, sem ter que colocar os valores e a palavra NOTE toda vez que tocarmos uma nota.

Como vamos definir um conjunto de palavras que serão muito úteis no momento em que quisermos utilizar música, introduziremos todas as definições seguintes no Editor, e guardaremos num arquivo sob o nome de "MUSIC-WORDS".

Inicialmente, especificaremos as oitavas através de uma variável, sabendo que uma freqüência de uma nota numa determinada escala difere da mesma nota numa escala diferente por um fator de potência de 2:

```
1  VARIABLE  OIT
:  1OIT   4  ->  OIT  ;
:  2OIT   2  ->  OIT  ;
:  3OIT   1  ->  OIT  ;
```

Devemos agora definir as possíveis durações das notas segundo as figuras definidas anteriormente, simplesmente armazenando numa variável (DUR) o valor da duração correspondente. Este procedimento facilita o trabalho, uma vez que, quando se repetem as durações em várias notas consecutivas, não há necessidade de se especificar a duração para cada uma delas, uma vez que ela é igual e já está armazenada na variável DUR.

```
4  VARIABLE  DUR
:  SB   255  ->  DUR  ;
:   M   128  ->  DUR  ;
:  SM    64  ->  DUR  ;
```



```
:   C   32  ->  DUR  ;
:  SC   16  ->  DUR  ;
:   F    8  ->  DUR  ;
:  SF    4  ->  DUR  ;
```

As notas vão ser montadas de modo a deixar na pilha o valor da altura correspondente àquela oitava, deixar na pilha o valor da duração que já fora especificado a executar a palavra NOTE. Observe que apenas especificamos as notas da 3ª oitava, as outras são calculadas dividindo-as pelo fator de 2 armazenado em OIT:

```
:  DOH   208  OIT  /  DUR  NOTE  ;
:  DOH#  197  OIT  /  DUR  NOTE  ;
:  RE    186  OIT  /  DUR  NOTE  ;
:  RE#   175  OIT  /  DUR  NOTE  ;
:  MI    165  OIT  /  DUR  NOTE  ;
:  FA    156  OIT  /  DUR  NOTE  ;
:  FA#   147  OIT  /  DUR  NOTE  ;
:  SOL   139  OIT  /  DUR  NOTE  ;
:  SOL#  131  OIT  /  DUR  NOTE  ;
:  LA    124  OIT  /  DUR  NOTE  ;
:  LA#   117  OIT  /  DUR  NOTE  ;
:  SI    110  OIT  /  DUR  NOTE  ;
```

Desta forma, cobrimos todas as notas dentro das 3 oitavas definidas.

Podemos ainda definir o ponto de aumento que, quando aparecer após uma palavra de duração, aumenta esse valor em 50% (note que não pode ser usado com SB já que esse é o valor máximo).

```
:  .A  DUR  DUP  2  /  +  ->  DUR  ;
```

Além disso, especificamos a PAUSA que causa um intervalo de silêncio com a duração especificada pelas mesmas palavras que especificam a duração para as notas.

```
:  PAUSA  DUR  10  *  0  DO  LOOP  ;
```

Com isso, completamos as palavras que necessitávamos definir para poder executar qualquer música no GraFORTH, conhecendo-se o seu pentagrama.

Veja que para emitir uma nota, devemos especificar a oitava escolhida (enquanto não mudarmos de oitava, não há necessidade de reatualizar esse valor), a duração de nota e se ela é pontuada ou não e a nota propriamente dita. Como exemplo, vejamos como emitir a

nota RÉ na 2ª oitava, numa mínima e depois numa fusa pontuada, separadas por uma pausa de semicolcheia:

```
Ready 2OIT M RE SC PAUSA F .A RE
```

21.1 Convertendo o pentagrama

Com as codificações de pautas que demos na introdução e as conversões para valores numéricos acima, é extremamente simples traduzir qualquer pentagrama. O conhecimento básico necessário de teoria musical na programação de música já foi dado. A dificuldade na tradução será maior ou menor em proporção à dificuldade e complexidade da pauta. Qualquer pauta pode ser convertida, porém as mais simples são as escritas para flauta e acordeom.

Para exemplificar melhor esta tradução, apresentamos a seguir um trecho da partitura de uma música do folclore americano muito divulgada entre nós. Sobre ela estão marcadas as palavras correspondentes no GraFORTH às notas e durações de cada figura. Segue também a listagem da definição da música no GraFORTH.

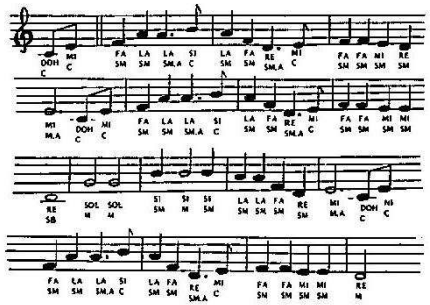


Figura 21.1 — Pentagrama de OH, SUZANA!

```
: OH.SUZANA
  2OIT C DOH MI SM FA LA .A LA
  C SI SM LA FA .A RE C MI SM
  FA FA MI RE M .A MI C DOH
  MI SM FA LA .A LA C SI SM LA
  FA .A RE C MI SM FA FA MI
  MI SB RE M SOL SOL SM SI M
  SI SM SI SM LA LA FA RE M
  .A MI C DOH MI SM FA LA .A
  LA C SI SM LA FA .A RE C MI
  SM FA FA MI MI M RE ;
```


21.2 Coleção de músicas

A seguir, apresentamos listagens de várias músicas tiradas de pentagramas, que podem ser armazenadas num arquivo e utilizadas em qualquer programa:

```
: LILIMARLENE
  2OIT SM MI C .A MI SC FA SM
  .A SOL C MI .A FA SC MI C
  .A FA 3OIT SC DOH 2OIT M SI
  SM RE C .A RE SC MI SM FA C
  .A FA SC SOL C .A SI SC LA
  C .A SOL SC FA M MI SM LA C
  .A SI 3OIT SC DOH 2OIT SM SI
  LA LA SOL .A SI C LA SM SOL
  FA .A LA C SOL SM FA MI .A
  SOL C MI SM .A SOL C FA SM
  FA 3OIT RE M DOH SM DOH 2OIT
  MI .A SOL C FA SM FA RE M
  DOH SM DOH ;

: REPT DOH# 3OIT SOL 2OIT ;

: NONA-SINFONIA
  2OIT SM DOH# DOH# RE MI MI RE
  DOH# 3OIT SI LA LA SI 2OIT
  DOH# DOH# 3OIT SI SI PAUSA
  2OIT DOH# DOH# RE MI MI RE
  DOH# 3OIT SI LA LA SI 2OIT
  DOH# 3OIT SI LA LA PAUSA SI SI
  2OIT DOH# 3OIT LA SI 2OIT RE
  DOH# 3OIT LA SI 2OIT RE DOH#
  3OIT SI LA SI LA ;

: O.GUARANI
  3OIT SM MI FA# SOL# LA SI LA
  LA LA SOL# FA# FA# FA# MI FA#
  PAUSA PAUSA MI SOL# SOL# SOL#
  LA SOL# SOL# SOL# MI RE RE RE
  MI MI ;

: JINGLE.BELLS
  SM 2OIT DOH# DOH# DOH# PAUSA
  DOH# DOH# DOH# PAUSA RE RE RE
  RE RE DOH# DOH# DOH# MI MI RE
  3OIT SI LA ;
```

```
: BRASILEIRINHO
  2OIT C FA FA FA FA DOH# 3OIT
  SOL# 2OIT FA FA REPT FA REPT
  FA REPT MI REPT MI REPT MI
  REPT MI REPT FA REPT FA REPT
  FA FA FA FA FA MI RE# RE DOH#
  3OIT SOL LA SI 2OIT DOH 3OIT
  SOL LA SI 2OIT DOH 3OIT SOL LA
  SI 2OIT DOH 3OIT SI LA SOL
  2OIT DOH# 3OIT SOL LA 2OIT DOH
  DOH# 3OIT SOL LA 2OIT DOH DOH#
  DOH 3OIT LA SOL 2OIT DOH 3OIT
  SOL LA SI 2OIT DOH 3OIT SOL
  SOL LA SOL FA# MI RE DOH# ;

: VALSA.DO.ADEUS
  SM 3OIT MI LA PAUSA PAUSA LA
  LA PAUSA 2OIT DOH# 3OIT PAUSA
  SI PAULA PAUSA LA SI PAUSA
  2OIT DOH# 3OIT PAUSA LA PAUSA
  PAUSA LA 2OIT DOH# PAUSA MI
  PAUSA FA# PAUSA PAUSA PAUSA
  PAUSA FA# PAUSA MI PAUSA PAUSA
  DOH# DOH# PAUSA 3OIT LA PAUSA
  SI PAUSA PAUSA LA SI PAUSA
  2OIT DOH# ;

: ATIREI.O.PAU.NO.GATO
  SM 2OIT MI PAUSA PAUSA RE DOH#
  3OIT SI 2OIT RE MI PAUSA MI
  PAUSA MI PAUSA FA# MI RE PAUSA
  RE PAUSA RE PAUSA MI RE DOH#
  PAUSA DOH# PAUSA DOH# PAUSA 3OIT
  LA LA 2OIT FA# PAUSA FA# PAUSA
  FA# PAUSA SOL# FA# MI PAUSA
  DOH# RE MI RE DOH# 3OIT SI
  LA ;
```

21.3 A palavra VOICE

Como foi dito no capítulo anterior, o sistema GraFORTH permite selecionar 9 "vozes" diferentes, através da palavra VOICE. Na realidade, cada "voz" seleciona um nível de volume e intensidade no qual as notas são emitidas. São apresentados os valores possíveis para a palavra VOICE:

-6 a -1	:	Seleciona um ciclo constante em volume. -1 = 50% do ciclo, -2 = 25% do ciclo, -3 = 12,5% do ciclo, e assim por diante. Um ciclo menor diminui o volume.
0	:	A nota inicia com um ciclo em 50% e decresce para 0%.
1	:	A nota inicia com 0%, aumenta para 50% e decresce para 0% novamente.
2	:	A nota começa em 0% e aumenta para 50%.

Ao carregar-se inicialmente o sistema, o valor assumido para VOICE é zero.

Aconselhamos que você experimente todos os valores possíveis de VOICE com uma determinada música, para poder ouvir e sentir bem a diferença em se usar um ou outro valor.

Por exemplo:

```
Ready 1 VOICE BRASILEIRINHO
Ready -3 VOICE O.GUARANI
Ready -1 VOICE O.GUARANI
Ready 2 VOICE JINGLE.BELLS
```

21.4 Conclusões

Este capítulo abordou basicamente os conceitos de partituras, ou seja, músicas escritas. Não apresentamos efeitos sonoros, tais como bombas, naves, raio laser ou coisas do gênero, por serem derivados de sons que podem ser produzidos da mesma maneira empregada para as notas musicais – duração e freqüência. Use sua imaginação.

Num dos próximos capítulos, onde vamos aplicar os conceitos gráficos com exemplos de desenvolvimento de aplicativos, também mostraremos como incrementar programas com sonorização.

Para finalizar, entre com o seguinte trecho musical e tente identificar a que música ele pertence:

```
: TRECHO
  3OIT SM DOH C .A FA SC MI C
  .A FA SC SOL C .A LA SC SOL
  C .A LA SC SI M .A SI 2OIT
  C DOH 3OIT SM FA DOH C .A FA
```

```
SC  MI  C  .A  SOL  SC  FA  C  .A
LA  SC  SOL  C  .A  LA#  SC  LA  SM
FA#  SOL  PAUSA  DOH  C  .A  SOL  SC
FA#  C  .A  SOL  SC  LA  C  .A  LA#
SC  LA  C  .A  LA#  SC  2OIT  DOH
SM  .A  DOH#  C  RE  3OIT  SM  SOL
DOH  C  .A  SOL  SC  FA#  C  .A  LA
SC  SOL  C  .A  LA#  SC  LA  C  .A
2OIT  DOH  3OIT  SC  LA#  SM  SOL#
LA  ;
```


Parte oitava

Animação tridimensional

22

PRINCÍPIOS BÁSICOS DOS GRÁFICOS TRIDIMENSIONAIS

22.0 Introdução

O sistema GraFORTH foi desenvolvido especialmente para poder manipular figuras no espaço tridimensional e para isso teve de vencer uma grande dificuldade: a lentidão do processamento. Fica claro desde já que se tentássemos reescrever as rotinas do GraFORTH que manipulam gráficos em Applesoft BASIC, por exemplo, não só não conseguiríamos pela falta de recursos desta última linguagem, como seria impossível termos a sensação de movimento, devido à baixíssima velocidade de interpretação dos comandos.

Com o GraFORTH, as imagens conseguem atingir velocidades de processamento comparáveis aos programas escritos em assembly, e também devido a algumas restrições, fazendo com que esse problema de atraso entre duas imagens consecutivas fosse minimizado.

Nesta seção vamos apresentar, basicamente, as palavras definidas no sistema para manipular até 16 figuras tridimensionais simultaneamente, e os editores que o auxiliarão na criação e movimentação dessas figuras, exemplificando, quando necessário, com as figuras já definidas que acompanham o sistema. Deixamos, para o próximo capítulo, a descrição completa dos passos para a elaboração de uma animação tridimensional completa e inédita.

Mostraremos também a estruturação do arquivo de imagem, ou seja, como a formação da imagem tridimensional é armazenada na memória, as tabelas de valores matemáticos de transformação – como escala, rotação e translação – e os catálogos dos arquivos de imagem.

22.1 Trabalhando com imagens tridimensionais

Inicialmente, vamos supor que já possuímos uma figura tridimensional cuja "imagem numérica" está guardada no disquete do sistema (posteriormente, quando analisarmos os editores, veremos como criar novas figuras).

As palavras do GraFORTH para manipular essas figuras são:

Quadro 22.1 **Palavras de manipulação de gráficos tridimensionais**

SCREEN	DRAW	SEQUENCE	UNDRAW	AUTODRAW
OBJECT	OBJADR	OBJERASE	OBJCOLOR	SCALE
SCALX	SCALY	SCALZ	XPOS	YPOS
XTRAN	YTRAN	ZTRAN	XROT	YROT
ZROT	OFF			

Essas palavras, simplesmente, dizem ao sistema onde as figuras estão armazenadas na memória e como elas devem ser exibidas na tela. Neste capítulo, explicaremos o funcionamento de cada uma dessas palavras.

Uma vez que temos um arquivo de imagens (representação numérica das linhas que formam o contorno do desenho) em disco, devemos carregá-lo na memória para utilizá-lo numa área livre (nossa velha conhecida, iniciando no endereço 2816). Existem várias figuras gravadas no sistema, mas vamos carregar a mais didática no momento:

```
Ready  CR 132 PUTC PRINT " BLOAD XYZ,A2816 " CR
```

Com este nome sugestivo, XYZ, esta figura nada mais é do que três setas perpendiculares entre si, representando os três eixos, cada uma de uma cor.

Porém, antes de exibir essa imagem na tela é bom reservar a parte inferior da tela para os comandos e mensagens alfanuméricas através de uma janela de texto.

```
Ready  ABORT
Ready  0  40  20  24  WINDOW  ERASE
```

Antes de mostrar "XYZ", devemos nos precaver de inicializar as tabelas de valores de transformação, quando iniciamos a manipulação de figuras tridimensionais, para não corrermos o risco de que os valores deixados por outras figuras danifiquem a imagem. Para isso, temos a palavra OBJERASE, de uso simples e sem parâmetros.

```
Ready  OBJERASE
```

Posteriormente indicaremos os valores com que os parâmetros são inicializados com o uso desta palavra.

Assim feito, podemos também estabelecer que o nosso desenho seja apresentado na tela toda vez que uma das palavras gráficas mostradas no início do capítulo seja executada. Isso é conseguido através da palavra AUTODRAW que espera, na pilha, um valor: se for 1, estabelece que o objeto correntemente selecionado deve ser desenhado após a execução de uma palavra gráfica qualquer; se for 0, desliga esse estado de "desenho automático". Assim, entre:

```
Ready  1  AUTODRAW
```

Estando o objeto carregado e definido na memória, devemos estabelecer os parâmetros que indicam a utilização deste objeto no momento, já que podem existir vários objetos na memória e o computador não tem meios de saber qual será mostrado no momento. Desta forma, devemos numerar os objetos e estabelecer o endereço de início de seu arquivo de imagem na memória. Assim, toda vez que nos referirmos a um objeto, basta designar o seu número através da palavra OBJECT. O sistema GraFORTH estabelece que todos os comandos gráficos são relacionados com esse objeto e que seu arquivo de imagem está num endereço já estabelecido e guardado numa tabela específica para aquele objeto, até que um novo objeto seja selecionado.

```
Ready  0  OBJET  2816  OBJADR
```

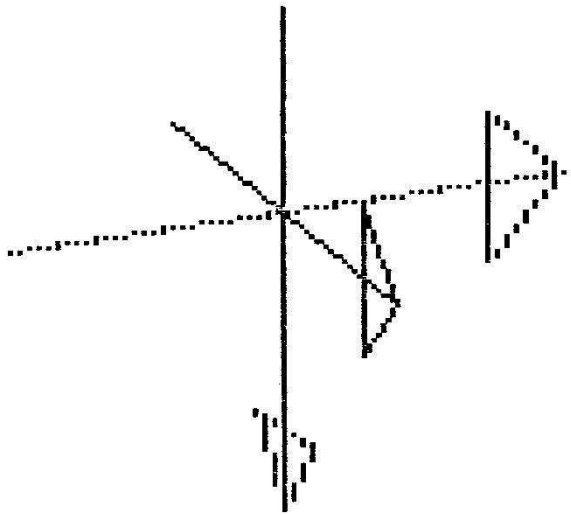


Figura 22.1 — As coordenadas XYZ.

As instruções citadas estabelecem que o nosso "XYZ" será o objeto 0 e, até novo uso da palavra OBJECT, todos os comandos gráficos seguintes se referem a esse objeto. Observe o surgimento do desenho na tela, devido à especificação do AUTODRAW. Observe que você está vendo apenas 2 dos eixos, uma vez que só é apresentado um plano para você, ou seja, uma vista ortogonal frontal do desenho. Porém, com os comandos que serão apresentados a seguir, os de rotação, escala e translação, poderemos ter uma visão espacial dos três eixos. Repare também que especificamos o endereço no qual está armazenado o arquivo de imagem deste objeto através da palavra OBJADR. Deve ficar claro que o objeto número zero está armazenado no endereço 2816. Porém, se colocarmos, por exemplo, 2890 OBJADR sem alterar o número do objeto, o sistema GraFORTH estabelece este novo endereço para o objeto zero. Todas as palavras gráficas se referem ao objeto zero até que este seja alterado através da palavra OBJECT.

22.2 O espaço tridimensional

Antes de introduzir as palavras que realmente movimentam as figuras tridimensionais, devemos apresentar as características de como são apresentadas, na tela, essas figuras.

Usamos, como no caso de figuras bidimensionais, o sistema cartesiano. Porém, agora necessitamos de três coordenadas: x, y e z, ao invés de apenas x e y, para localizar um ponto na tela. Diferentemente do caso bidimensional, a origem das coordenadas não está localizada no canto superior esquerdo da tela, nem tampouco noutro lugar fixo. Pois, a origem do sistema cartesiano é específica para cada figura, ou seja, quando mais de um objeto é exibido na tela, cada um deles possui sua própria origem de coordenadas, podendo ser posicionada em qualquer lugar na tela.

A faixa de valores permitidos para cada coordenada varia entre -128 e 127. Esta é uma diferença básica com relação às figuras bidimensionais, que não permitem coordenadas negativas. Note que para economia de espaço é usado apenas um byte para especificar o valor de uma coordenada. Porém, isto não significa que, por exemplo, uma linha reta vertical de comprimento 256 — desde (0, -128, 0) até (0, 127, 0) — ocupe toda a tela. O comprimento real apresentado na tela depende da escala utilizada. Por exemplo, na escala 0 para esse eixo, nada da reta será exibido na tela.

Apresentamos a seguir um diagrama onde o interior do cubo especifica o espaço que pode ser utilizado para a figura, onde variam os valores das coordenadas de -128 a 127:

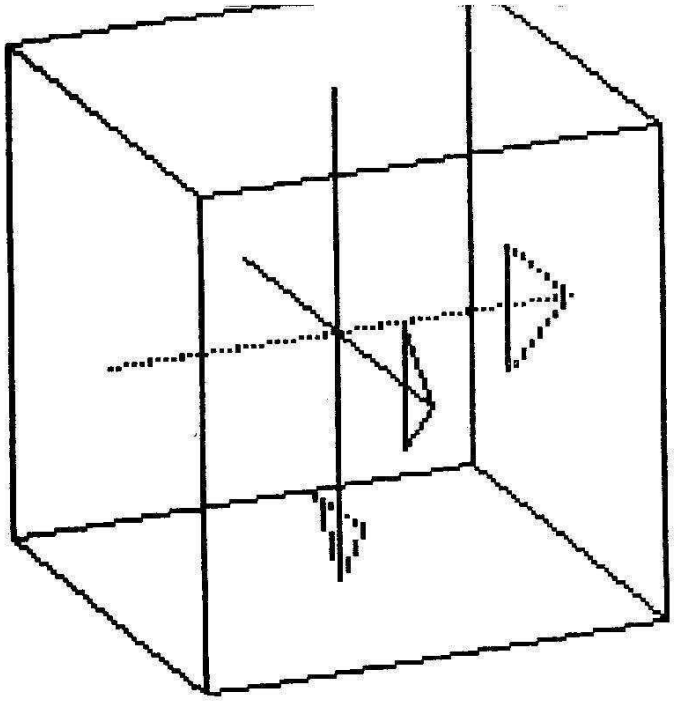


Figura 22.2 — Espaço tridimensional disponível.

22.3 Representação matricial de operações gráficas

A intenção destes parágrafos é de apresentar sucintamente os conceitos matemáticos envolvidos na manipulação (movimentação, rotação, escalonamento) de figuras num espaço tridimensional, do qual o sistema GraFORTH faz uso. Uma forma de representar as transformações gráficas pode ser através de equações matemáticas. Porém, esta não é adequada em termos de visualização das partes, além de ser de difícil manipulação computacionalmente falando, em comparação com a representação matricial. Os programadores sabem das facilidades que se têm em representar conjuntos de dados através de matrizes, tornando a elaboração de um trabalho muito mais simples.

No caso de transformações geométricas usando matrizes, os conceitos matemáticos mais avançados são necessários, tal como o conhecimento da aritmética matricial. Não vamos nos deter nesses detalhes por não fazerem parte do objetivo do livro, nem tampouco serem de maior interesse dos programadores e usuários do sistema GraFORTH, aos quais basta conhecer o "como usar".

No espaço tridimensional, um ponto necessita de três coordenadas (x, y e z) para ser localizado. Em notação matricial, podemos representá-lo por um vetor linha:

$$[\ x\ y\ z\ ] = P1$$

22.4 Translação em três dimensões

Dentro do espaço tridimensional, podemos movimentar um ponto acrescentando os valores da translação a cada uma das coordenadas. Representamos o vetor deslocamento, cujos termos representam o quanto será deslocado em determinado eixo, como sendo [Dx Dy Dz] = D e o processo de translação se resume a:

$$P1' = P1 + D \iff [\ x'\ y'\ z'\ ] = [\ x\ y\ z\ ] + [\ Dx\ Dy\ Dz\ ]$$

22.5 Fator de escala em três dimensões

Se estivermos usando um escalonamento uniforme em todas as direções, o vetor ponto é multiplicado pelo valor da escala, ou seja:

$$P1' = ESCALA * P1 \iff [\ x'\ y'\ z'\ ] = ESCALA * [\ x\ y\ z\ ]$$

Porém, se formos atribuir um escalonamento independente em cada direção, devemos multiplicar cada componente pelo seu respectivo fator de escala:

$$[\ x'\ y'\ z'\ ] = [\ ESCALAX * x \quad ESCALAY * y \quad ESCALAZ * z\ ]$$

Podemos colocar os fatores de escala na seguinte forma matricial:

$$Sxyz = \begin{bmatrix} ESCALAx & 0 & 0 \\ 0 & ESCALAy & 0 \\ 0 & 0 & ESCALAz \end{bmatrix}$$

Se fizermos a multiplicação, obteremos a expressão:

$$[\ x'\ y'\ z'\ ] = [\ x\ y\ z\ ] * \begin{bmatrix} ESCALAx & 0 & 0 \\ 0 & ESCALAy & 0 \\ 0 & 0 & ESCALAz \end{bmatrix}$$

ou $P1' = P1 * Sxyz$

22.6 Rotação em três dimensões

A forma mais simples de rotação de um elemento é aquela em torno de um dos eixos (x, y ou z). Através de uma dessas rotações, um determinado ponto se deslocará no espaço para uma nova posição, que pode ser determinada através das transformações matriciais abaixo:

Rotação em torno do eixo x:

$$[\ x'\ y'\ z'\ ] = [\ x\ y\ z\ ] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & COS\ \hat{x} & -SIN\ \hat{x} \\ 0 & SIN\ \hat{x} & COS\ \hat{x} \end{bmatrix}$$

Rotação em torno do eixo y:

$$[\ x'\ y'\ z'\ ] = [\ x\ y\ z\ ] * \begin{bmatrix} COS\ \hat{y} & 0 & -SIN\ \hat{y} \\ 0 & 1 & 0 \\ SIN\ \hat{y} & 0 & COS\ \hat{y} \end{bmatrix}$$

Rotação em torno do eixo z:

$$[\ x'\ y'\ z'\ ] = [\ x\ y\ z\ ] * \begin{bmatrix} COS\ \hat{z} & -SIN\ \hat{z} & 0 \\ SIN\ \hat{z} & COS\ \hat{z} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Essas matrizes de rotação assumem que a origem e o centro da rotação são tais como assumidos no GraFORTH. Porém, as rotações em torno de eixos paralelos aos eixos da origem podem ser possíveis pela translação dos eixos paralelos até a origem, executando a rotação e refazendo a translação inicial:

$$P1 = CR + (P1 - CR) * ROT3D$$

onde CR é o centro da rotação e ROT3D é uma das matrizes de rotação, ou uma multiplicação de várias delas, apresentadas acima.

22.7 Perspectiva

A perspectiva é uma forma de projeção que considera a noção de profundidade mais familiar ao modo como vemos os objetos na realidade, permitindo o efeito "ponto de fuga".

A idéia contida é que os raios de luz emitidos por um objeto entram radialmente nos olhos do observador. O que se representa na tela do computador é a intersecção desses raios de luz que chegam aos olhos do observador com um plano perpendicular a um desses raios, plano esse que é o plano da tela. A figura abaixo mostra essa representação e ilustra a seqüência de dedução das relações matemáticas:

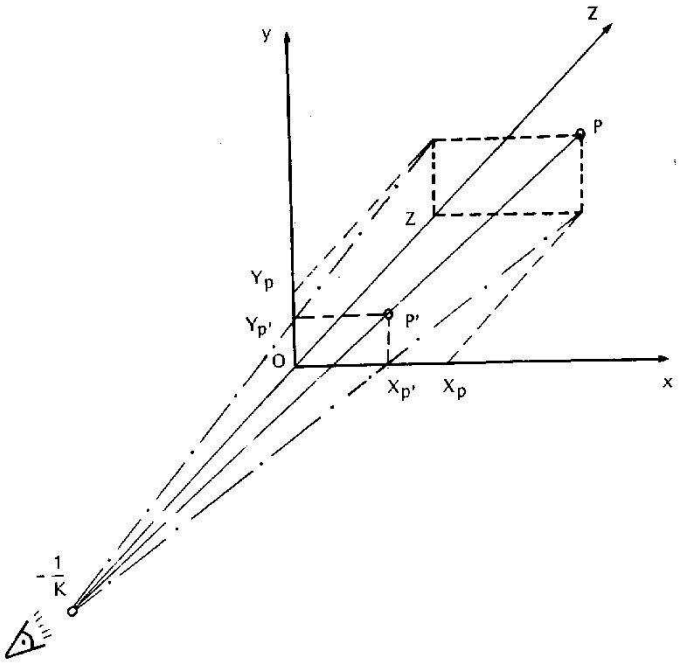


Figura 22.3 – Determinação geométrica da perspectiva de um ponto P.

Nesta figura, temos o observador colocado no ponto [0 0 -1/K] e um ponto P genérico [x y z] e a sua projeção no plano z = 0. Assim, os pontos [x' y' 0] representam os pontos da tela do monitor de vídeo do seu computador.

Usando de semelhança de triângulos na figura, obtemos:

$$\frac{x'}{1/K} = \frac{x}{z + 1/K} \Rightarrow x' = \frac{x}{K * z + 1}$$

e analogamente:

$$y' = \frac{y}{K * z + 1}$$

O que, em notação matricial, representa:

$$[\,x'\ y'\ z'\,] = [\,x\ y\ z\,] * \begin{bmatrix} 1/(K*z+1) & 0 & 0 \\ 0 & 1/(K*z+1) & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Notemos que quando 1/K aumenta, K se aproxima de zero e, no limite, temos uma projeção ortogonal (onde os raios de luz chegam ao olho do observador sob trajetórias paralelas: é o caso de objetos muito distantes para os quais não se tem a noção de profundidade).

22.8 Armazenamento e formação de imagens tridimensionais

Como já dissemos, uma figura tridimensional é armazenada na memória como uma seqüência de bytes que representam as coordenadas espaciais dos pontos, que ligados formam a representação da figura na tela. Cada ponto é armazenado como uma seqüência de quatro bytes. O primeiro é um byte indicativo: se o bit 7 estiver setado, a linha que liga o ponto anterior a este deve ser traçada, caso contrário, não. Os bits 0 a 2 representam o número do código da cor a ser usada na linha (ao invés da cor especificada para o objeto como um todo, através de OBJCOLOR que será melhor explicado mais adiante). Os bits 3, 4, 5 e 6 não são utilizados.

Os três bytes seguintes correspondem às coordenadas x, y e z do ponto dentro do espaço.

O fim do arquivo de imagem é indicado pelo primeiro byte igual a 255 ($FF).

Além dessa descrição, o sistema GraFORTH necessita guardar os atributos de cada objeto. Para isso, ele usa as seguintes áreas, iniciando nos seguintes endereços, como tabelas de atributos dos objetos:

5888 ($1700)
6144 ($1800)
6400 ($1900)

Cada área ocupa 256 bytes e contém 16 tabelas, uma para cada um dos 16 objetos possíveis de serem manipulados ao mesmo tempo, tendo cada uma delas 16 bytes nos quais são mantidos os seguintes atributos:

BYTE	ATRIBUTO
0	Flag (desenhar ou não desenhar).
1	Ângulo de rotação em torno de x.
2	Ângulo de rotação em torno de y.
3	Ângulo de rotação em torno de z.
4	Translação na direção x.
5	Translação na direção y.
6	Translação na direção z.
7	Posição x na tela.
8	Posição y na tela.
9	Fator de escala em x.
10	Fator de escala em y.
11	Fator de perspectiva.
12	Cor do objeto.
13 a 14	Endereço da imagem.
15	Não utilizado.

Note que cada tabela começa com um endereço múltiplo de 16, sendo fácil localizar qualquer atributo de um objeto. Por exemplo, para localizarmos a posição y na tela da origem do objeto 6, basta multiplicar 16 pelo número do objeto (6), adicionar a posição relativa na tabela do atributo desejado (8) e somar o endereço de início da área de tabelas (6400), no caso obtendo o endereço 6504.

Além dessas regiões, são mantidas, na página zero do Apple II, as matrizes de transformações geométricas necessárias para a rotação, translação e escalonamento dos objetos.

Elas são sucessivamente multiplicadas para obter a matriz de transformação equivalente que é aplicada a cada linha a ser exibida.

As quatro matrizes analisadas no item 22.3 são:

MATRIZ 1:
$$\begin{bmatrix} ESCALAx & 0 & 0 \\ 0 & ESCALAy & 0 \\ 0 & 0 & ESCALAz \end{bmatrix}$$

MATRIZ 2:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & COS(x) & -SEN(x) \\ 0 & SEN(x) & COS(x) \end{bmatrix}$$

MATRIZ 3:
$$\begin{bmatrix} COS(y) & 0 & -SEN(y) \\ 0 & 1 & 0 \\ SEN(y) & 0 & COS(y) \end{bmatrix}$$

MATRIZ 4:
$$\begin{bmatrix} COS(z) & -SEN(z) & 0 \\ SEN(z) & COS(z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A matriz resultante de transformação é uma para cada objeto, de modo que a multiplicação só necessita ser feita uma vez para o mesmo objeto. Então, o resultado é utilizado para transformar cada linha e depois são adicionados os termos de translação em cada coordenada. Depois é verificado se o valor de escala em z é diferente de zero e calculada a perspectiva do objeto, o que acarreta num tempo de processamento 20% maior. Finalmente são desenhadas, na tela, as retas que formam o desenho a partir da posição especificada para a origem (XPOS e YPOS).

23

RECURSOS DE ANIMAÇÃO DO GRAFORTH

23.0 Rotação

O sistema GraFORTH provê palavras que permitem especificar os valores para os termos usados nas matrizes de transformação, como os valores dos ângulos de rotação em torno dos eixos. São elas: XROT, YROT e ZROT. Todas esperam na pilha o valor do ângulo para o qual será rotacionado o objeto. Os valores dos ângulos variam na faixa de 0 a 256, sendo que 256 corresponde a 360 graus, ou uma revolução completa; os valores intermediários são proporcionais a essa relação. Por exemplo, 128 corresponde a 180 graus, 64 corresponde a 90 graus e 32 a 45 graus.

Deve ficar claro que o valor especifica o ângulo final no qual o desenho será feito, e não um incremento no valor atual para "rodar mais tantos graus em torno de x (ou y)". Se especificarmos 0 YROT, por exemplo, estamos estabelecendo que o plano xy é frontal ao observador, 90 YROT torna o plano zy frontal ao observador e 256 YROT retorna o plano xy (256 é o mesmo que 0, pois corresponde a uma volta completa).

Se você ainda mantém, na memória, o arquivo da imagem do "XYZ" e o AUTODRAW ainda está estabelecido (lembre-se de que as palavras de rotação XROT, YROT e ZROT são gráficas e afetam a AUTODRAW, produzindo o desenho de uma figura na tela a cada vez que uma delas é executada) podemos observar alguns efeitos. Entre com o seguinte programa:

```
: YSPIN
  14 XROT
  260 0 DO
        I YROT
  4 LOOP ;
  Ready YSPIN
```

Observe os eixos girando ao redor do eixo y. Você pode observar os efeitos semelhantes com XROT e ZROT.

23.1 Escala

Existem quatro palavras no GraFORTH que manipulam a forma e o tamanho com que um objeto é apresentado. São elas: SCALE, SCALX, SCALY e SCALZ. Todas elas esperam um valor na pilha dentro da faixa -31 a +31. Os números fora desta faixa são convertidos para dentro dessa faixa (da mesma forma que 375 graus correspondem a 15 graus). Quando as matrizes e as tabelas dos objetos são inicializadas através de OBJERASE, o valor 16 é atribuído a SCALX e SCALY e 0 para SCALZ.

A função de SCALX é alterar o comprimento do objeto. O valor zero torna o objeto comprimido a uma linha e com valores negativos obtém-se a imagem invertida. O processo é muito útil quando se trata de figuras que são imagens especulares de outras. Quando só é necessário definir uma e outra imagem, emprega-se uma escala negativa.

Como exemplo, execute o seguinte programa:

```
: ' SQUASH
  12 -12 DO
        I SCALX
  LOOP ;

  Ready 14 YROT SQUASH
```

As observações acima também são válidas para SCALY, porém com alteração da altura do objeto.

As palavras SCALX e SCALY são usadas quando se quer obter valores diferentes para altura e comprimento. Se, porém, você quiser uma escala homogênea com valores iguais para SCALX e SCALY, pode-se usar a palavra SCALE, que faz exatamente isso: atribuir o valor no topo da pilha para SCALX e SCALY.

A palavra SCALZ é responsável pelo termo de perspectiva com que é visto o objeto na tela. Corresponde ao termo K na matriz de perspectiva analisada anteriormente.

Como foi dito, o valor zero para SCALZ não promove perspectiva e o tempo de execução da imagem é 20% menor. Neste caso, observa-se que a parte frontal do objeto tem o mesmo tamanho da parte posterior.

Se for atribuído um valor positivo para SCALZ, obtém-se uma projeção perspectiva onde a parte frontal do objeto é maior que a posterior, e é tão significativa essa diferença quanto maior for o valor da escala.

Ocorre o mesmo para um valor negativo, porém, com a parte posterior maior que a anterior.

Verifique com os seguintes exemplos:

```
Ready  25  SCALZ  YSPIN
Ready  -15  SCALZ  YSPIN
Ready  0  SCALZ  YSPIN
```

Note também como é um pouco mais demorada a execução da palavra YSPIN para os valores de SCALZ diferentes de zero.

23.2 Posição

As palavras XPOS e YPOS são utilizadas para se especificar em que posição na tela deve ser colocada a origem do sistema de coordenadas do espaço. Ambos esperam na pilha um valor, sendo que XPOS espera a coordenada x (com um valor variando entre 0 e 255) e YPOS, a coordenada y (com um valor variando entre 0 e 191). Os valores atribuídos pela palavra OBJERASE são 128 para XPOS e 96 para YPOS, correspondendo ao centro da tela.

Ao se movimentar o objeto pela tela, pode ocorrer que parte dele seja repartida para o canto oposto da tela, causando linhas traçadas atravessando a tela, desfigurando completamente o objeto inicial, fenômeno denominado "wrap-around". Para que tal não ocorra, é necessário tornar pequena a figura (com um valor pequeno para SCALE) e evitar jogar o objeto muito para os cantos da tela.

Observe você mesmo, entrando:

```
Ready  6  SCALE
Ready  50  XPOS
Ready  40  YPOS
Ready  250  XPOS
Ready  5  YPOS
Ready  96  YPOS  128  XPOS
```

23.3 Translação

A translação difere da posição pelo fato de o objeto se mover dentro do seu próprio espaço tridimensional, enquanto XPOS e YPOS movimentam a origem do espaço do objeto na tela.

Os objetos podem ser transladados nos três eixos através das palavras XTRAN, YTRAN e ZTRAN que esperam na pilha o valor da nova coordenada da origem. Por exemplo, se um objeto tem um de seus pontos com as coordenadas [-30, 120, 0] e executarmos 65 XTRAN, o ponto terá as novas coordenadas [35, 120, 0] (pois a origem passou do [0, 0, 0] para o ponto [65, 0, 0]).

Deve-se tomar cuidado para que o valor final das coordenadas dos pontos esteja dentro dos valores permitidos para as coordenadas (de -128 a +127), pois, caso contrário, ocorrerá "wrap-around", mesmo que o objeto esteja numa escala pequena e não saia da tela, visto que isso ocorre dentro do próprio espaço tridimensional.

Exemplifiquemos para maior clareza. Suponha o ponto apresentado anteriormente com as coordenadas [-30, 120, 0]. Se transladarmos, em y, 50 posições, ou seja, entrarmos 50 YTRAN, ocorre o seguinte: 120 + 50 = 170, porém o valor máximo é 127, ou seja 128 corresponde a -128. Assim, basta subtrair 256 do resultado (170) para obter o valor final. As coordenadas do ponto seriam [-30, -86, 0].

Deste modo, fica claro porque ocorre o "wrap-around" dentro do próprio espaço do objeto. Isto pode ser evitado se definirmos as coordenadas do objeto com valores pequenos, longe dos limites, e não aplicarmos grandes translações.

No caso do arquivo "XYZ", não podemos transladar sem que o fenômeno ocorra, pois os eixos ocupam toda a faixa permitida (-128 a 127).

Para poder exemplificar o uso de translação, vamos carregar outro arquivo de imagem tridimensional, que não preenchendo todo o espaço, possa ser transladada. Vamos carregar a imagem "HOUSE" de uma casa, numa área livre da memória e ainda mantendo "XYZ".

```
Ready  ERASE
Ready  CR 132 PUTC PRINT " BLOAD HOUSE,A3000 " CR
Ready  1  OBJECT  3000 OBJADR
```

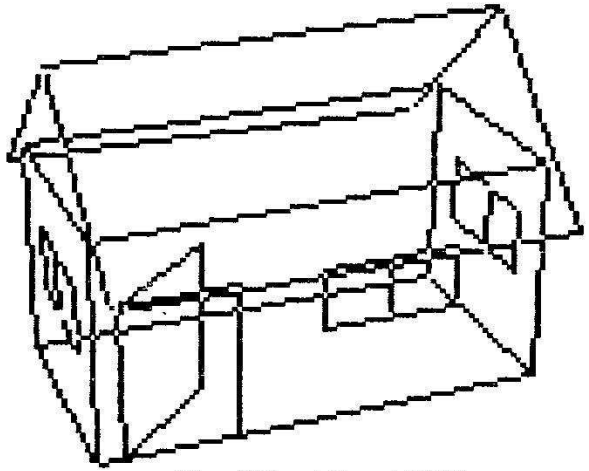


Figura 23.0 – A figura HOUSE.

Aparecerá a imagem da casa na tela. Observe as vistas que vão se formando com a seqüência de comandos apresentados:

```
Ready  20  XROT
Ready  10  YROT
Ready  8  SCALZ
Ready  10  SCALE
Ready  -50  ZTRAN
Ready  20  0  DO  I  ZTRAN  LOOP
Ready  50  ZTRAN
Ready  YSPIN
```

23.4 Cor

Quem possui um televisor ou monitor de vídeo colorido, pode observar que cada um dos eixos do "XYZ" possuía uma cor diferente. Isto quer dizer que quando foi montado o arquivo "XYZ", foi estabelecida uma cor especificada para cada eixo. Porém existia a possibilidade de não ser especificada nenhuma cor. A cor a ser utilizada é atribuída (e alterada facilmente) no momento de exibição do objeto através da palavra OBJCOLOR.

A casa não tem uma cor atribuída a seus elementos, assim você pode escolher a cor:

```
Ready  1  OBJCOLOR
Ready  3  OBJCOLOR
Ready  6  OBJCOLOR
```

Da mesma forma como você aprendeu a usar os gráficos bidimensionais e os caracteres gráficos, os modos INVERSE ou NORMAL, EXMODE ou ORMODE podem ser utilizados com os gráficos tridimensionais, produzindo interessantes efeitos.

23.5 Métodos de apresentação das imagens

Até agora, sempre que executamos uma palavra gráfica, o resultado da operação era mostrado na tela, pois havíamos ativado o modo AUTODRAW. Porém, dentro de um programa, usualmente, usamos a palavra DRAW, de modo que podemos alterar vários atributos antes que a nova imagem seja produzida.

Quando executamos a palavra DRAW, as rotinas gráficas (que traçam linhas, plotam pontos etc.) são dirigidas para a página de alta resolução que não está sendo mostrada naquele momento, de modo

que o processo de desenho não pode ser visto. A imagem anterior, que estava nessa página de texto que não está sendo mostrada, é apagada através dos comandos gráficos UNLINE e UNPLOT, a partir de dados que foram armazenados quando essa figura foi desenhada. Então a nova imagem é desenhada a partir dos atributos que foram especificados e alterados desde o último comando DRAW. É trocada a página de alta resolução que está sendo mostrada, de modo que se pode agora ver a nova imagem. Esse método garante uma imagem e animação de alta qualidade, visto que todo o processo de montagem do objeto é executado fora da visão do usuário.

Os caracteres gráficos discutidos nos capítulos anteriores podem ser facilmente mixados a imagens tridimensionais, uma vez que eles são desenhados em ambas as páginas de alta resolução.

Quando você tem uma cena com vários objetos, nem todos podem estar se movimentando num determinado momento. Assim, o comando DRAW só vai desenhar, dos objetos presentes na cena, os que foram referenciados (através da palavra OBJECT) desde o último DRAW. Desta forma, elimina-se o tempo de desenho de figuras que permanecem inalteradas em seus atributos. Exemplo:

```
0  OBJECT  20  XTRAN  15  YROT
2  OBJECT   5  SCALE   4  SCALZ
100  XPOS  DRAW
```

O comando DRAW só redesenhará, com os novos parâmetros, os objetos 0 e 2 que foram referenciados. Outros quaisquer que sejam mostrados na tela permanecem inalterados.

Pelo que foi dito, deve ter ficado claro que existem imagens diferentes do mesmo objeto nas duas páginas de alta resolução: a versão mais atualizada está na página que está sendo mostrada e a versão anterior na outra página. O que pode ocorrer, quando existem vários objetos sendo mostrados numa cena e repetidamente alternamos as páginas (através de sucessivos DRAW num dado objeto), é que as versões diferentes dos outros objetos não referenciados vão se alternando na tela – juntamente com a alternância das páginas de alta resolução – causando uma movimentação residual prejudicial à cena.

Podemos eliminar esse problema simplesmente referenciando esses objetos com imagens diferentes uma vez, para que ambas as imagens em ambas as páginas se tornem iguais. Por exemplo:

```
1  OBJECT
```

Seguido do próximo DRAW, porém sem nenhuma alteração nos atributos, causará uma repetição da imagem na outra página, deixando as duas páginas com a mesma imagem.

No caso em que se têm vários objetos se sobrepondo na tela, é aconselhável usar EXMODE para impedir que os mesmos sejam destruídos durante a execução da palavra DRAW. O mesmo é recomendado se os objetos se sobrepõem a caracteres.

Podemos querer simplesmente apagar determinado objeto. Você deve estar lembrado que quando se explicou o processo de apresentação de uma imagem, a anterior que estava presente na página de alta resolução e que não estava sendo mostrada era apagada através de UNLINE e UNPLOT. Muito bem. Podemos indicar ao sistema que após ter apagado o objeto, não faça mais nada, ou seja, não desenhe o objeto na nova posição. Isto equivale a apagar o objeto. Isto é feito usando-se a palavra OFF após a referência do objeto:

```
3 OBJECT OFF
```

Esta seqüência de palavras causará o apagamento do objeto 3 na próxima execução de um comando DRAW.

Explicamos que um objeto é apagado através do comando DRAW por ser desenhado no caminho inverso, pelo uso de UNLINE e UNPLOT. Porém, podemos apagar o objeto de uma maneira mais rápida. Basta que saibamos qual a área, em termos de caracteres, que ele ocupa e apagamos toda essa área (note que nessa área só deve existir o objeto a ser apagado ou redesenhado, pois se houver outros objetos ou caracteres, também serão apagados). É o que UNBLK faz. Porém se usarmos UNBLK para apagar a região e depois usarmos DRAW para redesenhar o objeto, perder-se-á tempo do mesmo modo, pois tentamos apagar algo que já foi apagado.

Para resolver esse problema, foi criada a palavra UNDRAW, que funciona exatamente da mesma forma que UNBLK, só que é avisado que ele não necessita apagar o objeto antigo antes de redesenhar, conseguindo-se assim uma maior velocidade de processamento.

Vamos explicar melhor o funcionamento de UNDRAW através de um exemplo hipotético. Imagine um objeto posicionado no centro da tela, com seus pontos todos interiores a um círculo de 20 pontos de raio. A área coberta por UNDRAW (da forma como é estabelecida por BLKSIZE) é medida através de caracteres, sendo que cada caractere possui 7 pontos de largura por 8 de altura. Assim, escolhendo um bloco de 6 por 5 caracteres, cobrimos uma área de 42 pontos de largura por 40 de altura, que corresponde à área do objeto. Devemos localizar essa região através de VTAB e HTAB de modo que fique centralizada na tela, na mesma posição onde se encontra o objeto, executando um UNDRAW antes do próximo DRAW. Resumindo:

```
6 5 BLKSIZE          ( estabelece o tamanho da área a ser usada )
                     ( pela palavra UNDRAW )
```



```
18 VTAB 17 HTAB      ( centraliza o bloco na tela )
UNDRAW DRAW          ( apaga o bloco e redesenha o objeto mais )
                     ( rapidamente que um simples DRAW )
```

Assim como UNBLK e PUTBLK, UNDRAW não altera VTAB e HTAB após a impressão, de forma que se você quiser apagar sucessivamente a mesma região não há necessidade de reposicionar o bloco todas as vezes.

Pode-se também desejar que os objetos não sejam apagados antes de desenhá-los na nova posição. É um efeito interessante de sobreposição de imagens, como quando se deixa o diafragma da câmera aberto por um longo tempo e os objetos a serem fotografados são iluminados por uma luz estroboscópica (emitida a intervalos de tempo regulares).

O efeito é obtido simplesmente usando UNDRAW posicionando numa região da tela externa à usada pelo objeto. O tamanho do bloco, inclusive, pode ser 1 por 1 para aumento de velocidade.

Também é possível impedir a troca de páginas de alta resolução usadas pelo comando DRAW, obrigando a que os desenhos sejam feitos todos numa única página. Desta forma, é mostrado todo o processo de montagem do objeto.

A palavra no GraFORTH que faz tal operação é SEQUENCE, que espera na pilha um valor: se for zero, estabelece que não ocorrerá a troca de páginas quando do uso do comando DRAW; se for um, a troca será esfetuada.

Para estabelecer que página será exibida quando do uso de 0 SEQUENCE, existe a palavra SCREEN que espera um valor, 0 ou 1, referentes respectivamente à página um ou dois de alta resolução.

Para exemplificar o uso de UNDRAW, vamos definir YSPIN para o nosso objeto "XYZ" (os eixos), porém sem o uso de AUTODRAW, e sim DRAW:

```
Ready  0 AUTODRAW ERASE 0 40 1 24 WINDOW
Ready  0 OBJECT OBJERASE 2816 OBJADR
Ready  5 SCALE  16 XROT  14 YROT  DRAW
Ready  6  5 BLKSIZE
Ready  :  YSPIN2
          260  0  DO
                 1 YROT  UNDRAW  DRAW
          4     +LOOP  ;
Ready  10 VTAB  15 HTAB  YSPIN  2  18 VTAB
```

Entre o seguinte, depois de executar o programa acima:

```
Ready  1  AUTODRAW  YSPIN  0  AUTODRAW
```

Observe a diferença de velocidade de processamento das duas imagens. Experimente agora o seguinte:

```
Ready 1 1 BLKSIZE 0 SEQUENCE
      0 SCREEN 18 SCALE
Ready ERASE 1 VTAB 1 HTAB
      YSPIN2 18 VTAB
```

Aqui pode ser visto o efeito de sobreposição de imagens. Tente também no modo EXMODE:

```
Ready ERASE EXMODE YSPIN2 18 VTAB
```

Deste modo, demos uma descrição do funcionamento e da utilidade das palavras gráficas que havíamos apresentado no início do capítulo, juntamente com alguns pequenos exemplos de utilização. Os próximos capítulos tratarão unicamente de desenvolvimento e análise de programas, utilizando essas palavras gráficas e algumas técnicas empregadas, para o leitor se familiarizar melhor com as possibilidades do GraFORTH. A seguir analisamos um dos programas demonstrativos onde algumas das técnicas mostradas anteriormente são exploradas.

23.6 FLEDERMAUS

Um dos aplicativos que mais chama a atenção é a animação do morceguinho voando sobre um céu estrelado. Este programa demonstrativo possui algumas características que apresentamos anteriormente, e por isso o escolhemos para exemplificar os tipos de técnicas que podemos utilizar com gráficos tridimensionais.

Entrando no editor de textos do sistema GraFORTH, usando o comando G com o nome FLEDERMAUS, e seguido de L, obtemos a listagem do programa demonstrativo que será analisado.

A partir da linha 850 está a palavra BATSPIRAL que é a principal responsável pelo vôo do morcego.

Inicialmente, o tamanho do bloco é colocado 1 por 1 caractere no canto inferior esquerdo (técnica já explicada de observação de todo o processo de animação). Feito isso, o valor 256 é armazenado na pilha de retorno e inicia-se um loop, que se encerra caso seja pressionada alguma tecla ou esse valor no topo da pilha de retorno seja nulo (note que a palavra I sempre retorna ao topo da pilha de dados, o topo da pilha de retorno sem apagar esse valor).

O morcego é formado por 3 objetos: 2 asas e o corpo. O movimento do morcego é dado pela escolha correta de valores para os parâmetros de rotação, escala e posição, que são alterados continuamente dentro do loop do BEGIN-UNTIL. As palavras XP e YP se

Figura 23.1 – FLEDERMAUS.

encarregam de posicionar os 3 objetos que compõem o morcego, segundo as fórmulas matemáticas:

```
XP = 128 + 5 * SEN ( I + 32 ) / 6
YP = 148 - I / 2
```

onde I varia de 256 a 0. Isso equivale a dizer que a trajetória do morcego corresponde a uma senóide em torno da tela, no sentido vertical.

A palavra YR se encarrega de ir rodando o morcego, virando-o de frente e de lado conforme percorre sua trajetória senoidal, representada pela fórmula, também em função de I, que dá o ângulo para o giro:

```
YR = 2 * I
```

Esta fórmula corresponde a duas voltas, já que I varia de 255 a 0 (uma volta).

O fator de escala também varia, pois o morcego está afastado quando vai da direita para a esquerda (a 1ª meia-volta) e se aproxima quando vai da esquerda para a direita (a 2ª meia-volta), sendo as escalas mínimas e máximas, respectivamente, na metade de cada meia-volta. Matematicamente, pode ser representado por:

```
SC = 5 + SEN ( I ) / 50
```

onde SC é a palavra que realiza a escala dos três objetos que formam o morcego.

Para dar a noção do bater de asas, basta imaginar que o eixo z passa pelo corpo do morcego com as asas apontando para o eixo x. Assim, se girarmos uma asa de X graus em torno de z e a outra de 180-X graus, também em torno de Z, e fizermos X aumentar até um máximo, depois decrescer até um mínimo e repetir esse ciclo (função senoidal), teremos o movimento de bater as asas. No programa, é usado um valor maior que 180 graus para se obter um efeito de perspectiva.

```
Para a primeira asa (objeto 0):    ZROT = SEN ( I + 11 ) /·3
Para a segunda asa (objeto 1):     ZROT = 132 - SEN ( I + 11 ) / 3
```

Após ter todos os parâmetros atualizados, é verificado se desejamos que as figuras sejam sobrepostas sem que as antigas sejam apagadas (através de UNDRAW.FLAG = 1 que causa a execução do UNDRAW numa posição onde não existe nenhum objeto; observe que isso é utilizado na palavra LEFT.RIGHT.SHOW, onde o valor 1 é atribuído a UNDRAW.FLAG e 0 para SEQUENCE, por isso o "rastro" deixado pelos morcegos, na apresentação das espécies), ou se desejamos animação normal (efeito comum do comando DRAW).

Em seguida, o valor de I (topo da pilha de retorno) é decrementado para repetir o processo.

Antes, porém, para que a palavra BATSPIRAL tenha o efeito desejado, devemos zerar o flag UNDRAW.FLAG, estabelecer vídeo NORMAL ou INVERSO conforme a preferência (note que os dois modos são utilizados no demonstrativo), e estabelecer o modo EXMODE, com 1 SEQUENCE.

Esta análise serviu principalmente para exemplificar como se estabelece os valores dos parâmetros dos objetos quando se tem um movimento a descrever. A melhor maneira para se conseguir isso de uma forma aproximada é através de fórmulas matemáticas, onde um ou mais loops, alternando uma ou diversas variáveis, facilmente estabelecem uma continuidade de movimentos.

Num capítulo posterior, teremos outro exemplo, agora com o acompanhamento de todo o desenvolvimento, de uma outra animação tridimensional.

24

EDITORES DE IMAGENS TRIDIMENSIONAIS

24.0 Edição de imagens: IMAGEDITOR

Até aqui, estivemos mostrando como manipular imagens tridimensionais já prontas. Porém, não apresentamos meios de criar seus próprios objetos.

Para esse propósito, existe, no sistema GraFORTH, um conjunto de editores de imagem tridimensional (assim como o CHAREDITOR serve para a criação de caracteres), e particularmente há um editor chamado IMAGEDITOR.

Antes de carregar o editor, você deve ter um arquivo de imagem ("XYZ") na memória para aprender a manipular os comandos.

IMAGEDITOR é um programa extremamente extenso e ocupará a região destinada ao editor de texto, caso seu sistema Apple II não possua um "language card" ou uma placa expansão de memória. Deste modo, sugerimos que você a esqueça (FORGET) o IMAGE DITOR antes de utilizar o editor de texto. Agora, para carregar o IMAGEDITOR, é aconselhável também esquecer todas as palavras novas que foram inseridas no topo do dicionário.

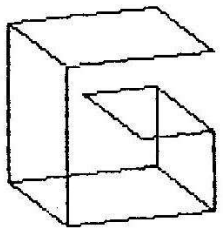
```
Ready  ABORT
Ready  READ  " IMAGEDITOR "
Ready  RUN
```

Agora, você está dentro do IMAGEDITOR e pode ver uma lista de comandos no lado direito e a mensagem "Enter Command:" aguardando que algum comando seja dado. Você está, então, apto a criar ou editar qualquer imagem tridimensional manipulável pelo sistema GraFORTH.

A seguir, apresentaremos todos os comandos do IMAGEDITOR na ordem em que eles aparecem no menu.

24.0.0 – A – Address

mensagem: Enter file address: 2816



```
A=Address
G=Get
K=Keep
R=Rotate
P=Position
S=Scale
W=Window
L=List
C=Color
M=Mode
E=Enter
Z=Zero
Q=Quit

OC X    Y    Z    Addr
M  100  100  100   876
D  100  100 -100    80
M -100  100  100   884
D -100  100 -100    88
M -100 -100  100   892
D -100 -100 -100    96
(M)ove,(D)raw,(-)Delete,(CR)Quit :
```

Figura 24.0 – Tela do Imageditor.

Este comando deve ser usado para indicar ao editor em que endereço da memória se inicia o arquivo de imagem do objeto a ser usado. Independe se existe ou não um objeto nesse endereço. O endereço mais comumente usado é apresentado juntamente com a mensagem, bastando passar o cursor por sobre os dígitos e pressionar a tecla ⟨RETURN⟩, se desejar este valor. Deve ser claro para você que somente um objeto pode ser manipulado por vez no IMAGEDITOR. Para mudar de objeto, deve-se alterar o endereço. Note também que não é mostrado o objeto na tela até que seja usado um dos comandos de movimentação (rotação, escala, translação). *Nota:* ao invés de 2816, poderá aparecer, na mensagem, o último endereço especificado por este comando.

24.0.1 – G – Get
mensagem: Enter file name to get:

Comando utilizado para trazer, para a memória, um arquivo de imagem armazenado no disquete. Após a mensagem, deve ser introduzido o nome do arquivo no disquete. O arquivo será armazenado no endereço especificado no comando A.

24.0.2 – K – Keep
mensagem: Enter file name to keep:

Usado para salvar, em disquete, o arquivo de imagem em que estamos trabalhando, com o nome introduzido após a mensagem.

24.0.3 – R – Rotate
mensagem: Rotate [X(num) to Z(num)]:

Comando usado para promover rotação do objeto em torno de um dos três eixos coordenados. A mensagem pede para que seja entrada uma letra correspondente ao eixo em que se quer a rotação (x, y ou z), seguido do valor numérico do ângulo final de rotação. Lembre-se de que os valores dos ângulos estão na faixa 0-256, correspondendo 256 a um ângulo de 360 graus.

24.0.4 – P – Position
mensagem: Position [X(num) or Y(num)]:

Do mesmo modo, este comando espera, após a mensagem, uma letra indicando um dos eixos da tela seguido do valor numérico, correspondendo à posição na tela onde deve ser colocada a origem da imagem. O valor inicial admitido, até alguma alteração, é X=64 e Y=48.

24.0.5 – S – Scale
mensagem: Scale [(num), or X,Y,Z (num)]:

Comando utilizado para alterar a escala de apresentação do desenho ou sua perspectiva. Dois modos de introdução de dados são possíveis: se você deseja a mesma escala tanto para a largura quanto para a altura, basta introduzir apenas um valor numérico para essas escalas. Se você deseja valores diferentes para cada um dos eixos, deve entrar a letra do eixo (x, y ou z) seguida do valor numérico da escala. Lembre-se de que os valores de escala estão na faixa -31 a +31.

24.0.6 – W – Window
mensagem: Enter text window top (0-23):

Comando utilizado para deslocar a janela de texto, aumentando ou diminuindo a área disponível para desenhos. Espera que você entre com o número da linha superior da janela de texto. Este comando também é usado para limpar a tela dentro do IMAGEDITOR, o que é feito todas as vezes que uma nova posição é setada.

24.0.7 – L – List

```
mensagem: OC    X    Y    Z    Addr
          ...   ...  ...  ...  .....
          (return = More, CTRL-C = Exit)
```

Comando utilizado para apresentar o conteúdo numérico do arquivo de imagem. O formato de apresentação consta de seis colunas: a primeira apresenta a letra (M) se este ponto em questão for somente de movimentação e a letra (D) se for para traçar uma linha desde o último ponto até este. A segunda apresenta o número da cor especificada ou nenhuma, caso seja usada a palavra OBJCOLOR para esse fim. A terceira, a quarta e a quinta colunas apresentam respectivamente os valores das coordenadas x, y e z, dentro do limite -128 a +127. A última coluna mostra o endereço inicial de cada registro do arquivo de imagem. O tamanho do registro é sempre de 4 bytes.

Este comando apresenta um certo número de linhas de cada vez, esperando que a tecla ⟨RETURN⟩ seja pressionada para continuar a listagem, ou que as teclas ⟨CTRL⟩⟨C⟩ encerrem o comando.

24.0.8 – C – Color

mensagem: Enter begin color (0-7):

Comando que especifica a cor com que deve ser apresentado o objeto na tela.

24.0.9 – M – Mode

mensagem: Mode (X) = Exclusive Or, (O) = Or:

Comando usado para especificar o modo. Se desejarmos EXMODE, devemos entrar com (X). Porém, se quisermos o modo ORMODE, entraremos com (O).

24.0.10 – E – Enter

mensagem: (M)ore, (D)raw, (-)Delete, (CR)quit:

Este é o comando principal do IMAGEDITOR, usado para criar novos objetos. Na realidade, ele permite que se entre com os valores numéricos de cada dado do arquivo de imagem. Lista, inclusive, as seis últimas linhas do arquivo.

Selecionando (M) ou (D) (dependendo se será ou não traçada uma reta), ele pedirá uma cor e os valores para as coordenadas x, y e z (pressionando ⟨RETURN⟩, serão aceitos os valores anteriores, sem a necessidade de repeti-los).

Selecionando (-), será apagada a última linha entrada.

A tecla ⟨RETURN⟩ abandona este comando.

24.0.11 – Z – Zero (Erase)

mensagem: Erase File (Y/N):

Comando usado para apagar completamente o arquivo de imagem, respondendo Y à mensagem.

É apagado o arquivo de imagem na memória, permanecendo na tela a última imagem mostrada. Se desejarmos que a tela seja limpa, usaremos o comando W.

24.0.12 – Q – Quit

Comando de retorno ao sistema GraFORTH, limpando toda a tela e inicializando as variáveis do sistema (corresponde à execução da palavra ABORT).

24.1 Edição de imagem: PROFILE

O sistema GraFORTH possui um outro editor de imagens, chamado PROFILE, que é específico para criar imagens de objetos de revolução, ou seja, imagens de objetos formados pela rotação de uma linha no espaço em torno de um eixo. O objeto tridimensional formado possui uma estrutura cilíndrica, que pode ser representada por uma imagem na tela via cortes perpendiculares ao eixo de rotação (que não passam de circunferências) e cortes passando por planos radiais contendo esse eixo.

Basicamente, o que o PROFILE faz é criar circunferências aproximadas por polígonos regulares contidos em planos paralelos ao plano xz, ao longo do eixo y, considerado o eixo de revolução, e em seguida ligando os vértices dos polígonos para formar a "casca" do objeto. Deste modo, necessitamos apenas especificar o número de lados do polígono que desejamos para aproximar as circunferências, a posição do centro da circunferência sobre o eixo y (coordenada y) e o valor do raio, correspondendo à coordenada x.

Existe, no disquete, o arquivo "CHAL" de um cálice que exemplifica muito bem o tipo de objetos que podem ser descritos com o editor PROFILE.

Para carregar o PROFILE, devemos eliminar, do dicionário, todas as palavras extras que foram inseridas a entrar:

```
Ready READ " PROFILE "
Ready RUN
```

```
[ Profile Generator ]

This routine generates a polygonal
object for GraFORTH ][ 3D graphics
when given a profile for the object.

Profile data are X,Y pairs in the
range -127, 127.

Enter number of polygon sides : 20

Enter Object File Address : 2816

Data from [K]eyboard or [D]isk ?
```

Figura 24.1 — Tela de apresentação do PROFILE.

O PROFILE foi escrito de tal forma que não existem vários comandos que possam ser acessados ao mesmo tempo, mas sim dirigindo as ações do usuário para que ele vá sistematicamente introduzindo os dados necessários.

Depois de uma breve apresentação, a primeira mensagem apresentada é a seguinte:

Enter number of polygon sides:

Aqui você é intimado a introduzir o número de lados do polígono regular que será usado para aproximar uma circunferência. Obviamente, quanto maior o número de lados, maior será a precisão da circunferência. Porém, maiores serão também o espaço de memória e o tempo de processamento gastos.

A seguir, pedir-se-lhe-á para especificar o endereço de armazenamento da imagem produzida.

Enter object file address:

Neste caso, pode-se usar o já conhecido endereço 2816. Porém, para imagens muito grandes, corre-se o risco dela invadir o sistema GraFORTH, perdendo-se assim o controle. Para evitar esses problemas, é aconselhável guardar o arquivo acima do topo do dicionário (que é obtido através de PRGTOP), onde o espaço é maior.

A seguir, o editor deseja saber se vai entrar com os dados a partir de um arquivo em disco criado pelo editor, ou se através do teclado.

Data from [K]eyboard or [D]isk?

Neste momento, se a opção [D]isk for escolhida, surgirá a mensagem:

Enter Data file name:

onde é solicitado o nome do arquivo de dados armazenado no disquete. Este arquivo é essencialmente uma seqüência de par de números, um em cada linha, separados por vírgula correspondente a X, Y, onde X é o raio (entre 0 e 127) e Y é a posição no eixo y (entre -128 e 127), finalizados pela letra E (END). O arquivo BIGCHAL é um exemplo de como deve ser organizado o arquivo de dados para o PROFILE.

Se ao invés de (D)isk, fosse escolhido (K)eyboard, teríamos de entrar, via teclado, os valores dos pares acima citados. A mensagem que surgiria é a seguinte:

Enter X,Y pair (end="E"):

Esta mensagem é repetida a cada par de raio, posição entrada, até que a letra E seja introduzida, indicando fim de dados.

Ao final desse processo, o PROFILE gera um arquivo de imagem, mostra-nos quantos bytes ocupam esse arquivo e desenha a imagem na tela.

No final, mais uma mensagem é exibida:

Enter object file name:

Esta mensagem nos pede o nome do arquivo do disquete onde será guardado o arquivo de imagem recém-criado. Porém, se nenhum nome for entrado e simplesmente se pressionou a tecla ⟨RETURN⟩, nada será gravado.

Os arquivos de imagens gerados pelo PROFILE são, em geral, muito grandes. Por esse motivo o editor deve verificar se o arquivo criado não vai ultrapassar áreas indevidas (como as do próprio sistema GraFORTH, as do DOS ou as de I/O do Apple II). Se isso ocorrer, nenhuma imagem é criada e a seguinte mensagem é exibida:

Not enough room here.
(Requires nnnn bytes.)

onde nnnn é o número de bytes requeridos pelo arquivo de imagem.

24.2 Edição de Imagem: PLAY

Este programa foi especialmente desenvolvido para facilitar a criação de animações, ou seja, seqüência de movimentos com os objetos tridimensionais criados pelos outros dois editores (IMAGEDITOR e PROFILE).

O programa é carregado do disquete, tendo-se os mesmos cuidados que se tinha com os outros dois editores: eliminar as palavras extras:

```
Ready READ " PLAY "
Ready RUN
```

Temos, inicialmente, uma tela de apresentação de comandos, que correspondem aos valores de atributos, tais como rotação, escala, translação e posição.

Você, inicialmente, seleciona a tecla correspondente ao atributo que você deseja alterar (veja a correspondência abaixo):

TECLA	ATRIBUTO
⟨1⟩	Rotação em X
⟨2⟩	Rotação em Y
⟨3⟩	Rotação em Z
⟨4⟩	Escala em X
⟨5⟩	Escala em Y
⟨6⟩	Escala em Z
⟨7⟩	Translação em X
⟨8⟩	Translação em Y
⟨9⟩	Translação em Z
⟨:⟩	Posição X
⟨-⟩	Posição Y

Na última linha da tela, é apresentado o valor atual do atributo escolhido e o valor do incremento (inicialmente, valendo zero) que continuamente é acrescentado ao atributo e corresponde a um redesenho do objeto na nova posição. Com as teclas de cursor (<— e —>), você vai aumentando ou diminuindo o valor do incremento, o que permite controlar as condições e a velocidade com que o desenho vai alterando sua posição.

Estes incrementos contínuos e redesenhos do objeto dão a noção de movimento. Estudando esses valores dos atributos e de quanto em quanto devem ser incrementados, obtém-se a velocidade desejada, se os valores permitirem que não ocorra o efeito de "wrap-around", onde parte do objeto sai fora dos limites da tela etc.

Tudo isso permite escolher convenientemente os valores dos atributos de forma a se obter a animação desejada.

Além dos comandos que selecionam os atributos, existem outros de controle:

TECLA	COMANDO
⟨F⟩	Congela a animação. Os valores atuais do atributo são mantidos e os do incremento são zerados.
⟨D⟩	Restaura, no atributo atual, o valor inicial do programa. O incremento é zerado.
⟨ESC⟩	Restaura, em todos os atributos, seus respectivos valores iniciais.
⟨CTRL⟩⟨S⟩	Causa uma parada na animação até que alguma tecla seja pressionada. Este comando mantém inalterado o valor do incremento.
⟨?⟩	Exibe a tela de instruções.
⟨Q⟩	Abandona o editor PLAY. Para voltar ao editor, basta entrar RUN.

Antes, porém, de iniciar a exibição da imagem, o programa necessita saber onde se encontra o arquivo de imagem. Deve-se, portanto, responder as questões na última linha da tela de instruções, que deseja saber se a imagem está no disquete [D] ou já está na memória [M]. Se for selecionada a imagem no disquete, é solicitado o endereço para onde deve ser carregada (normalmente, o endereço 2816) e o nome do arquivo. Após serem introduzidos todos os dados iniciais, você será solicitado para teclar ⟨RETURN⟩ assim que desejar iniciar o programa.

```
PLAY - A 3-D Image Manipulator

ROTATION   SCALE   TRANSLATION   POSITION
  X Y Z     X Y Z     X Y Z        X Y

Press top row keys to select parameter,
  then press
    [1] [2]    to set in motion
    [3]        to freeze motion
    [0]        to reset default

Or press
    [ESC]      to reset all parameters
    [CTRL] [S] to pause display
    [?]        to see these instructions
    [Q]        to quit.

Image in [M]emory or on [D]isk?
```

Figura 24.2.0 — A tela de apresentação do PLAY.

PLAY permite-lhe alterar também vários atributos simultaneamente: podemos observar uma alteração de posição de um objeto, juntamente com rotações nos eixos x e y, além de alterar os fatores de escala. Desta forma, o editor PLAY permite todas as alterações e valores possíveis para os atributos de um objeto.

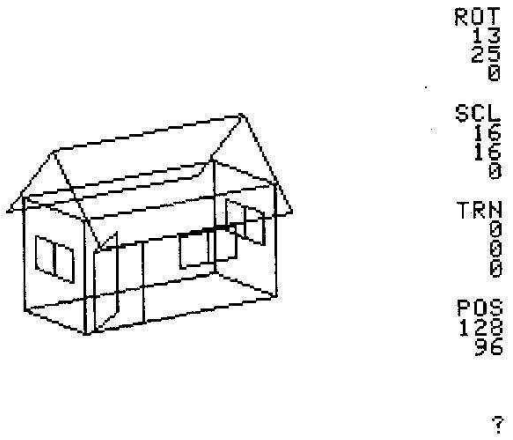


Figura 24.2.1 — Editando imagens tridimensionais com PLAY.

Parte nona

Programação e animação gráfica em GraFORTH

25

APLICANDO CONHECIMENTOS EM ANIMAÇÃO

25.0 Introdução

Nos capítulos anteriores, tivemos como preocupação básica apresentar os recursos disponíveis no sistema GraFORTH, inicialmente a nível de linguagem de programação e posteriormente a nível de comandos gráficos disponíveis e seus editores, sem contudo esgotar toda a potencialidade do sistema que apresenta apenas duas limitações: a memória do micro e sua imaginação.

Este capítulo tem como objetivo integrar todos os outros, ou seja, mostrar ao usuário como aplicar com eficiência, de forma organizada e precisa, os conhecimentos adquiridos e tentando induzir fluência na programação FORTH.

Apresentaremos programas-exemplos de aplicações, envolvendo caracteres gráficos e figuras tridimensionais, ilustrando princípios básicos de animação nestas áreas mais comuns.

Muitas pessoas, que se aventuram a aprender a linguagem FORTH, consideram-na inicialmente extremamente difícil de programar. Mais difícil ainda entender um programa escrito por outra pessoa. Mas esta opinião só é correta desde que o programa não seja bem planejado e estruturado, dividido convenientemente em módulos de fácil programação. Tivemos um cuidado muito especial nessa parte, tentando mostrar desde o início os passos para uma boa estruturação.

25.1 Desenvolvimento de um programa

De repente tem-se uma idéia original para um jogo que ninguém ainda tinha pensado, ou um utilitário que vai auxiliá-lo futuramente em muitas aplicações, reduzindo seu trabalho. Exatamente, esse é o início de todo bom programa, ou seja, a idéia do que se pretende obter.

O passo seguinte é o crucial. É a etapa que realmente decide se o programa vai ser bom ou não, fácil de programar ou não, fácil de corrigir ou não.

É o momento em que se deve definir cada passo do seu programa; suas funções, suas rotinas principais, o que pode, o que não pode e o como fazer.

Nessa etapa, a análise estruturada de um problema vem nos auxiliar para que o trabalho futuro na confecção do programa seja mínimo.

Através de uma organização funcional do programa, dividindo-o em módulos, que por sua vez são formados por rotinas, é que se consegue dividir as tarefas em tarefas menores, mais fáceis de se programar e menos propícias a erros. A maneira mais comum de se visualizar essa estrutura organizada de módulos relacionados entre si é conhecida como DHF (Diagrama Hierárquico de Fluxo). Exemplos de DHF serão vistos mais adiante.

Tendo-se conseguido esse diagrama, onde constam as rotinas básicas e os fluxos de dados entre as rotinas (quais os parâmetros de entrada de uma rotina e quais os de saída e sua localização na pilha), pode-se iniciar a etapa seguinte: a programação propriamente dita de cada módulo.

Para ficar mais claro, mais visual e didático, vamos acompanhar o que foi dito com exemplos.

25.2 ATAQUE ESPACIAL: um programa com caracteres gráficos

Como exemplo do uso de animação bidimensional, empregando a técnica da redefinição de caracteres, vamos fazer uma versão do mundialmente conhecido "SPACE INVADERS", aqui com o nome de ATAQUE ESPACIAL.

Já dissemos antes que a primeira e mais importante etapa é a de definir o programa que deverá ser feito. Primeiramente vamos descrever o jogo:

- Um conjunto de naves espaciais alienígenas e hostis se move em ziguezague pela tela, descendo até a sua base de defesa (última linha).
- O jogador possui uma base de defesa móvel que ele controla por dois comandos, um de movimentação à direita e outro de movimentação à esquerda, além de um terceiro comando de disparo.
- Somente você possui armas, as naves alienígenas estão desarmadas.
- Seu objetivo é destruir as naves antes que elas aterrissem (alcancem a última linha onde se encontra a sua base de defesa). Conseguindo detê-las, um novo conjunto de naves reaparecerá e o jogo reinicia-se. Pontos são acumulados no placar a cada nave destruída.

O primeiro passo é definir os tamanhos das naves e suas formas, redefinindo caracteres através do CHAREDITOR. Admitamos que tanto as naves quanto nossa base de defesa formem blocos de 3 por 2

caracteres. Além disso, os tiros terão o tamanho de um caractere. As naves terão dois formatos que se alternam com suas movimentações, para dar a impressão de movimento e tornar mais atraente o jogo. Usaremos ao todo 15 naves em 3 grupos de 5 e no máximo 10 tiros simultâneos.

Através do CHAREDITOR vamos definir as naves e o tiro que terão as formas esquematizadas abaixo, e que você deve reproduzir para a criação de um novo conjunto de caracteres que chamaremos de CHR.SPACE. Porém, antes de entrar no editor, não devemos nos esquecer de limpar a tela e "esquecer" as palavras desnecessárias do dicionário, cuidados estes já explicados anteriormente.

Uma vez dentro do CHAREDITOR, escolhemos a opção T (TRANSFER) para transferir os bytes de definição dos caracteres do sistema para o endereço 2816. Isto porque iremos definir as nossas figuras como caracteres, ficando os caracteres não usados preenchidos com os bytes normais de definição de caracteres. Usaremos a área livre que se inicia em 2816 para armazenar os novos caracteres (figuras). Assim respondemos 2048 para o endereço inicial e 2816 para o final, transferindo os 768 bytes que formam um conjunto de caracteres.

Pela opção A (ADDRESS), alteramos o endereço de trabalho para 2816, onde desejarmos. Com a opção D (DISPLAY) verificamos o sucesso da operação de transferência.

Podemos iniciar a criação dos desenhos. Inicialmente, vamos fazer uma das naves. Estabelecemos pelo comando B (BLOCK SIZE) o tamanho do bloco em 3 caracteres na horizontal por 2 na vertical.

Agora só nos resta passar o desenho abaixo definido (figura a) para a tela pelos comandos de movimentação I, J, K, M, P e L para marcar os pontos. Os comandos U e Z podem ser utilizados para corrigir eventuais erros.

Assim feito, é só passar a imagem definida na tela para o endereço correspondente no conjunto de caracteres que estamos criando, pelo comando W (WRITE). Ao sermos solicitados pelo número do caractere inicial do bloco respondemos 1.

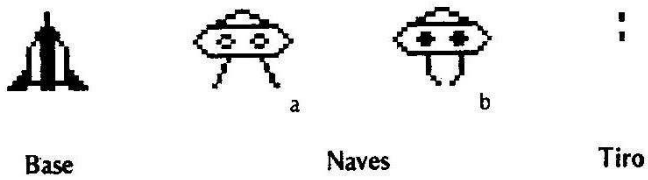


Figura 25.0 – Desenho de base, nave e TIRO

Os passos anteriores são repetidos para a outra nave (figura b), a nossa base e o tiro, sendo que o número inicial de cada um será 7, 13 e 21, respectivamente.

Tendo finalizado a definição dos objetos, teremos que salvar em disco esses caracteres, ou nosso trabalho terá sido em vão. Isso é feito pelo comando S (SAVE) e com o nome CHR.SPACE.

Note também que não vamos usar nenhum caractere para representar a explosão de uma nave (caractere com a nave em pedaços, por exemplo), pois esse efeito será obtido pela impressão da nave em modos INVERSE e NORMAL alternados.

Todas as figuras já estando criadas, vamos tratar de construir o programa. Vejamos o que ele deve ter como partes principais:

– ABERTURA: onde inicializamos variáveis, apresentamos o jogo, carregamos os conjuntos de caracteres necessários, estabelecemos os modos de operação etc.;

– PARTE PRINCIPAL: onde o jogo se desenvolve;

– PARTE FINAL: onde apagamos as palavras desnecessárias, estabelecemos o conjunto de caracteres do sistema e limpamos a tela.

Agora, tendo uma primeira divisão do problema, devemos analisar cada parte, subdividindo-as. Comecemos por ABERTURA, indicando as partes principais dessa rotina:

ABERTURA:

- limpar a tela;
- imprimir o nome do programa;
- carregar o conjunto de caracteres gráficos;
- inicializar as variáveis;
- estabelecer o tamanho do bloco.

Figura 25.1 – A tela de apresentação do Ataque.

De maneira geral pudemos definir a abertura, o processo de inicialização do jogo. Porém, uma melhor divisão – como quais são as variáveis que devemos inicializar – só será possível depois de termos bem claro como vamos executar o programa. Tentemos subdividir as tarefas do programa propriamente dito.

PARTE PRINCIPAL:

- movimentar objetos até acabarem as naves;
- diminuir o tempo de espera (DELAY);
- acrescentar pontos;
- reiniciar as variáveis;
- repetir as etapas acima até que seja destruído.

Teremos uma movimentação de naves mais lenta no início, tornando-se mais rápido no final, para aumentar a dificuldade. Isso é feito inserindo laços vazios no interior das rotinas, sendo que o valor máximo do contador desse laço deve variar, sendo decrementado ou incrementado conforme a necessidade. Esse valor máximo será armazenado na variável DELAY. Portanto, "diminuir o tempo de espera" significa diminuir o valor contido em DELAY.

Na maioria dos jogos, tem-se como objetivo marcar o maior número de pontos. É também o caso deste, onde armazenamos os pontos conquistados numa variável, PTOS, que é incrementada sempre que destruímos um OVNI, ou quando destruímos toda uma esquadrilha e passamos para uma nova fase. Interessante é que os pontos acumulados sejam variáveis; dependendo da situação ganham-se mais ou menos pontos.

A reinicialização das variáveis será vista mais tarde, quando tivermos todas as variáveis definidas. Daremos também a definição de toda a PARTE PRINCIPAL, assim que todas as palavras tenham sido definidas.

Precisamos ainda destrinchar melhor a movimentação dos objetos. Devemos deixar bem claro que temos três tipos de objetos a serem movimentados: naves, nossa base e tiros.

A partir deste instante vamos nos referir às naves alienígenas como OVNIs, nossa base como BASE e tiros como TIROS.

É interessante que a cada movimento de um dos objetos seja possível também movimentar os outros. Assim, devemos definir palavras para movimentar independentemente cada um dos objetos.

Como temos 15 OVNIs, se a cada movimento de um deles movimentássemos também a BASE e os TIROS, eles se moveriam muito mais lentamente do que os últimos. Assim, movimentaremos os

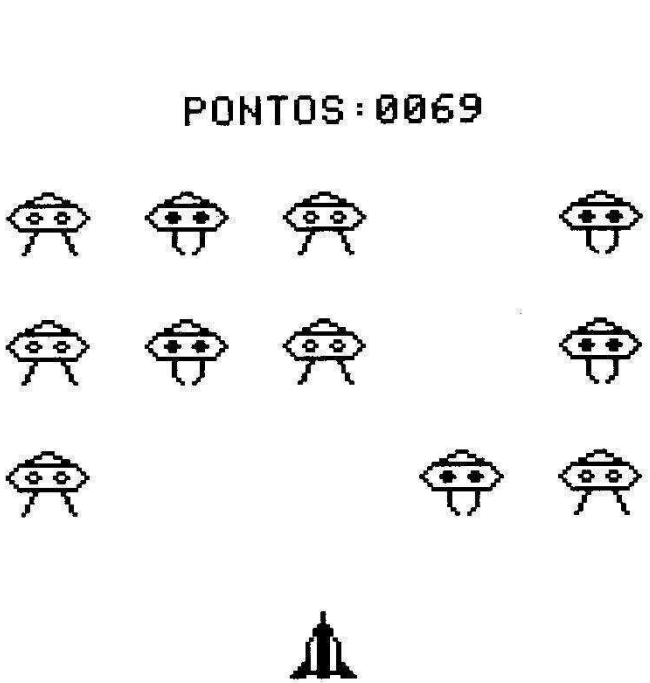


Figura 25.2 – A tela do jogo Ataque.

OVNIs em grupos de OVNIs pertencentes a uma mesma linha (no início 5 OVNIs). Devemos também especificar qual a direção do movimento dos OVNIs; eles se movem em ziguezague, ora para a direita, ora para a esquerda, descendo sempre que mudam de sentido.

Esse problema é resolvido armazenando-se as posições de cada OVNI (coordenadas x e y, correspondentes à coluna e à linha inicial de posicionamento do bloco). Como temos 15 OVNIs, para facilitar nosso acesso a essas informações, é aconselhável armazená-las em variáveis do tipo vetor ou matriz (em BASIC, A(I), B(I,J) etc.). Entretanto, não existem essas estruturas de dados definidas no GraFORTH. Porém, podem ser simuladas através de uma estrutura presente que é palavra-string.

Na definição de uma cadeia de caracteres, o sistema simplesmente reserva uma área de memória de tantos bytes quantos os especificados. Nesta área podemos não só armazenar caracteres como números de 1 byte, acessados através de PEEK e POKE. No nosso caso, definiremos XP e YP como variáveis strings onde armazenaremos as colunas e linhas de cada OVNI, cada uma com 15 posições. Por exemplo, para obtermos a coluna do primeiro OVNI, basta fazer

```
0 XP PEEK
```


e o valor será deixado na pilha. Para colocarmos o sexto OVNI na linha 21 basta fazer

```
21 6 YP POKE
```

Devemos agora observar o que acontece se um tiro atingir um OVNI. Este OVNI, obviamente, deixará de existir, sendo que não mais devemos movimentá-lo. Como diferenciar, em um dado instante, quais os OVNIs que foram destruídos e quais não foram? Isso também é facilmente resolvido, mantendo um flag para cada OVNI. Se o flag tiver o valor zero, o OVNI não existe mais, caso contrário o OVNI não foi destruído. Esse flag é definido também como uma variável string, da mesma forma que para XP e YP, de nome ATIVOS. No exemplo seguinte:

```
1 ATIVOS PEEK IF 1 XP PEEK . SPCE 1 YP PEEK . THEN
```

só serão impressos os valores das coordenadas do OVNI 1, se ele ainda estiver ATIVO.

Da mesma forma que para os OVNIs, também para os TIROS temos um flag para cada um dos 10 possíveis tiros (a STRING TIROS) e as coordenadas da BASE (variáveis XBASE e YBASE).

Deste modo, vejamos o que é feito na movimentação dos objetos:

MOVIMENTAÇÃO DOS OBJETOS

- movimenta os 5 primeiros OVNIs num sentido;
- movimenta TIROS;
- verifica se alguma tecla de comando foi pressionada;
- movimenta BASE;
- dispara um TIRO;
- verifica se algum OVNI foi atingido;
- repete os passos acima para os outros 2 grupos de OVNIs;
- repete os passos acima para mais 8 posições no sentido atual;
- decrementa a variável DELAY;
- movimenta todos os OVNIs para baixo;
- movimenta TIROS;
- verifica se alguma tecla de comando foi pressionada;
- movimenta BASE;
- dispara TIRO;
- verifica se algum OVNI foi atingido;
- repete os passos acima alternando o sentido, ora para a esquerda, ora para a direita;
- repete os passos acima até que você destrua todos os OVNIs ou eles cheguem até sua BASE.

A rotina de movimentação dos OVNIs deve ser a mais geral possível, pois a usamos para movimentar 5 ou 15 OVNIs, ora para a direita, para a esquerda, ora para baixo. Deste modo, vamos dar como parâmetros de entrada quatro valores:

incrementoY incrementoX OVNIfinal OVNIinicial MOV.OVNI

Os seguintes passos devem ser executados:

MOV.OVNI:

- verificar se o OVNI está ativo;
- apagar o OVNI da posição atual;
- somar incrementoX a XP;
- somar incrementoY a YP;
- desenhar o OVNI na nova posição;
- repetir desde OVNIinicial até OVNIfinal;
- verificar se a BASE foi destruída pelos OVNIs;
- explodir BASE caso isso tenha ocorrido.

Para definir essa palavra, ainda devemos subdividir melhor esses dois últimos passos:

VERIF.BASE:

- verifica se OVNI está ativo;
- verifica se YP+1=YBASE (o que significa que a BASE foi destruída);
- em caso afirmativo o flag DESTruído assume 1;
- repete para todos os OVNIs.

EXP.BASE:

- verifica se o flag DEST está ligado (DEST=1);
- em caso afirmativo EXPLODE a BASE.

EXPLODE:

- recebe como parâmetros a posição da explosão (XEXP e YEXP) e no topo da pilha o número do bloco do objeto a ser explodido (pode ser BASE ou OVNI);
- repete os passos seguintes 5 vezes;
- se for na vez par imprime em modo INVERSE;

- se for na vez ímpar imprime o modo NORMAL;
- posiciona e imprime o bloco correspondente;
- executa um laço de espera vazio, para se notar a alternância de inverso e normal;
- no final apaga o objeto e deixa em modo NORMAL.

A rotina de movimentação de tiros é semelhante à de movimentação de OVNIs, sendo que a verificação se o tiro atingiu algum OVNI é feita em rotina à parte. Somente deve-se movimentar o tiro se ele ainda não saiu do limite superior da tela, que para o jogo é a linha 2 (a linha 0 está ocupada com o placar e a 1 é mantida livre por precaução).

A movimentação da BASE é feita por sua reimpressão na nova posição, à direita ou à esquerda da posição anterior, caso a tecla correspondente tenha sido pressionada. Devido ao fato de mover à direita ser a mesma coisa que mover à esquerda, só que decrementando uma unidade na coordenada horizontal ao invés de acrescentar uma unidade, podemos definir uma palavra que execute essa movimentação, só que esperando na pilha o valor do incremento a ser dado, que pode ser positivo (mover à direita) ou negativo (mover à esquerda). Assim, a palavra MOV.BASE, que corresponde ao módulo principal deste procedimento, executaria os seguintes passos:

MOV.BASE:

- coloca incremento negativo na pilha (-1);
- verifica se é comando de movimentação à esquerda;
- e se XBASE atingiu a primeira coluna;
- executa POSIC.BASE;
- coloca incremento positivo na pilha (+1);
- verifica se é comando de movimentação à direita;
- e se XBASE atingiu a última coluna;
- executa POSIC.BASE;
- executa laço de espera.

A rotina que realmente desloca a nave numa dada direção é POSIC.BASE, onde devem ser executados os seguintes passos:

POSIC.BASE:

- apaga BASE da posição atual;
- some XBASE com o incremento e coloque em XBASE;
- imprime a BASE na nova posição.

Vejamos os passos necessários quando um tiro for disparado, que corresponde a tornar um tiro ativo na posição da BASE no momento do disparo:

NOVO.TIRO:

- verificar se o código é 224 (espaço);
- procurar dentre os 10 TIROs um que não esteja ATIVO;
- caso todos estejam ATIVOS, o último assume este novo TIRO;
- colocar nas coordenadas do TIRO, a coluna da BASE+1 (para dar a impressão que o TIRO sai do meio da BASE) e a linha da BASE -1;
- passar esse TIRO a ser ATIVO.

Devemos também verificar se algum dos OVNIs foi atingido pelos TIROS disparados. Um TIRO terá acertado um OVNI se a posição do TIRO corresponder a uma das seis posições dos seis caracteres de que é formado um OVNI. Corresponde a testarmos para todos os TIROs ativos e para todos os OVNIs ativos se a seguinte relação é verdadeira:

((YP+1) >= YT) E (YP <= YT) E ((XP+2) >= XT) E (XP<=XT)

onde XP e YP são as coordenadas do OVNI e
XT e YT são as coordenadas do TIRO.

Se essa expressão for verdadeira, então o OVNI foi atingido e deve ser eliminado. Para isso basta que zeremos a variável string ATIVOS, correspondente a este OVNI, zeremos também a variável string TIROS do TIRO correspondente; chamamos a rotina EXPLODE para explodir o OVNI e acrescentamos pontos, verificando se acabaram os OVNIs ou ainda existe algum. No caso, criamos uma variável para servir de flag, VENC, que assume o valor 1 caso todos os OVNIs de uma esquadrilha (grupos de 15 OVNIs) tenham sido destruídos, e zero caso contrário.

De uma forma geral, estão definidos todos os módulos do programa e podemos apresentar o DHF correspondente:

Podemos observar no DHF os retângulos que correspondem aos módulos nos quais se subdivide o programa, organizados hierarquicamente de cima para baixo. Os módulos ligados logo abaixo de um outro pertencem a este último, podendo no entanto ser usados em outros módulos. Se um mesmo módulo é chamado mais de uma vez, não é necessário que se repitam os módulos a ele subordinados.

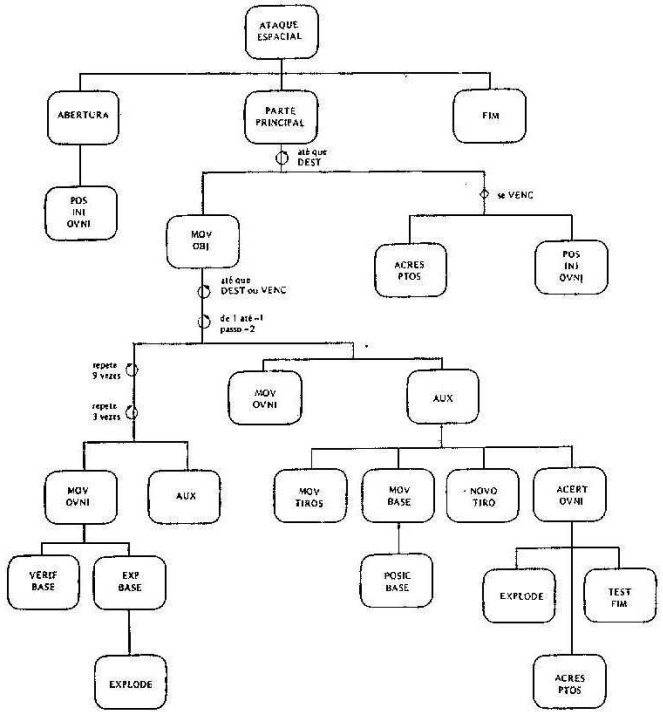


Figura 25.3 – DHF de ATAQUE ESPACIAL.

Uma vez tendo a descrição de todos os módulos do nosso programa, podemos passar a definir as palavras neles contidas, usadas neste jogo, acompanhadas de comentários servindo única e exclusivamente para esclarecer a lógica do programa. No final do capítulo, apresentamos uma listagem completa no formato adequado para se introduzir no EDITOR. É aconselhável também, para uma melhor compreensão, que se acompanhem as descrições dos módulos feitas anteriormente.

Primeiramente apresentamos a rotina de contabilização de pontos:

```
: ACRES.PTOS
   CHRSET CHRADR     ( passa para caracteres normais )
   15 + PTOS +       ( o valor do topo da pilha +15+ PTOS )
   DUP -> PTOS       ( é armazenado em PTOS e )
   0 VTAB 19 HTAB .  ( impresso na tela na primeira linha )
   2816 CHRADR ;     ( e os caracteres gráficos são restaurados )
```

A seguir se encontram as definições dos módulos da palavra de movimentação dos OVNIs, MOV.OVNI:

```
: EXPLODE
  5 0 DO                  ( repete 5 vezes )
    I 2 MOD 0 =           ( verifica se é par ou ímpar )
  IF INVERSE              ( par INVERSE )
  ELSE NORMAL             ( ímpar NORMAL, ocorre a alternância )
  THEN
  -936 CALL               ( necessário limpar a página de texto )
  XEXP HTAB               ( posiciona a coluna )
  YEXP VTAB               ( posiciona a linha )
  DUP PUTBLK              ( imprime o bloco cujo número está no )
    150 0 DO LOOP         ( topo da pilha. Executa loop vazio )
  LOOP                    ( de espera )
  NORMAL UNBLK            ( estabelece modo normal, apaga objeto )
  DROP ;                  ( e o número do objeto na pilha )
```

É bom relembrar que o sistema GraFORTH não imprime na tela de alta resolução um caractere já existente na página de texto, e deste modo não observaríamos nada. A solução encontrada é apagar a página de texto antes de reimprimir o bloco, daí o uso de -936 CALL.

```
: EXP.BASE
  DEST                    ( verifica se BASE foi substituída )
  IF XBASE -> XEXP        ( em caso afirmativo coloca a )
     YBASE -> YEXP        ( posição atual da nave nas )
                          ( variáveis usadas por )
     13 EXPLODE           ( EXPLODE e explode a base (13) )
  THEN ;

: VERIF.BASE
  15 0 DO                 ( repete para todos os OVNIs )
    I ATIVOS PEEK         ( se estiver ativo )
    IF I YP PEEK 1 +      ( YP+1 = YBASE )
       YBASE =
       DEST OR            ( ou já foi destruído por outro )
    -> DEST               ( então acende flag )
  THEN
  LOOP ;

: MOV.OVNI
  DO I ATIVOS PEEK        ( repete para os OVNIs indicados )
                          ( na pilha )
    IF I YP PEEK VTAB     ( se ATIVOS, posiciona linha e )
```



```
    I XP PEEK HTAB        ( colunas atuais )
    UNBLK                 ( apaga bloco )
    DUP I XP PEEK +       ( incrementa a coluna com o in- )
    DUP I XP POKE         ( cremento na pilha; atualiza XP )
    HTAB                  ( e posiciona )
    2 PICK I YP PEEK +    ( incrementa a linha )
    DUP I YP POKE         ( atualiza YP )
    VTAB                  ( e posiciona )
    N.OVNI PUTBLK         ( imprime o OVNI na nova posição )
  THEN
  DELAY 0 DO LOOP         ( laço vazio de espera )
LOOP
DROP DROP                 ( apaga os incrementos deixados )
VERIF.BASE                ( na pilha; verifica se BASE foi )
EXP.BASE ;                ( atingida; destrói em caso )
                          ( afirmativo )
```

N.OVNI é uma palavra que devolve na pilha, alternadamente, um dos dois números dos blocos de definição para os OVNIs (1 e 7). Segue a sua definição, que deve aparecer antes da definição de MOV. OVNI:

```
: N.OVNI
    1 6 N * +             ( N é um flag que se altera )
    N 0 = -> N ;          ( N.OVNI = 1+6*N, com N alternando-se )
                          ( entre 0 e 1 )
```

onde quando N=0 deixa o valor 1 na pilha e N passa a valer 1, e quando N=1 deixa 7 na pilha e N passa a valer zero.

Na definição da palavra MOV.TIROS notamos a semelhança com MOV.OVNI, de forma que os comentários são suficientes como explicação do funcionamento desta palavra:

```
: MOV.TIROS
  10 0 DO                 ( repete para todos os tiros )
    I TIROS PEEK          ( verifica se existe este TIRO )
    IF I YT PEEK 1 -      ( verifica se a próxima linha do )
       DUP 2 <=           ( TIRO é menor ou igual a 2 )
     IF 0 I TIROS POKE    ( se for desativa este TIRO )
       I YT PEEK VTAB     ( posiciona linha )
       I XT PEEK HTAB     ( posiciona coluna )
       UNBLK              ( apaga o TIRO da tela )
       DROP               ( apaga a nova linha da pilha )
     ELSE DUP             ( se a linha é maior que 2 )
       I YT POKE          ( atualiza YT )
```

```
      I XT PEEK
      2 PICK 1 + VTAB      ( posiciona a linha anterior, YT+1 )
      DUP HTAB             ( posiciona a coluna XT )
      UNBLK                ( apaga TIRO da posição anterior )
      HTAB VTAB            ( coloca na nova posição )
      21 PUTBLK            ( imprime TIRO na nova posição )
    THEN
  THEN
  DELAY 2 / 0              ( laço vazio com DELAY/2 )
  DO LOOP
LOOP ;
```

Neste jogo, temos três teclas de controle: duas de movimentação da BASE e uma de disparo. *A priori* poderemos escolher quaisquer teclas ou combinação de teclas para executar essas funções, bastando que se saiba o código ASCII correspondente.

Vamos escolher, para movimentação à direita, a seta correspondente, [—>], cujo código é 149; para a movimentação à esquerda a outra seta, [<—], cujo código é 136; e para o disparo a tecla de espaço, de código 224.

Imaginemos, inicialmente, como movimentar a BASE num dado sentido (direito ou esquerdo, simplesmente acrescentando um valor positivo ou negativo, respectivamente, para cada caso, a variável de posição correspondente à linha onde ela se encontra, XBASE). Isso é feito da mesma forma que nas movimentações anteriores, ou seja, apagando a posição anterior e redesenhando na nova. Este é o princípio básico do processo de animação que estamos querendo enfatizar.

```
: POSIC.BASE
   IF                  ( verifica se pode movimentar )
     XBASE             ( posiciona coordenadas atuais )
     DUP HTAB
     YBASE VTAB
     UNBLK             ( apaga da posição atual )
     PULL + DUP        ( retira da pilha de retorno o incremento; )
     —> XBASE          ( atualiza coluna de BASE )
     HTAB              ( coloca na nova posição )
     13 PUTBLK         ( imprime a BASE )
   ELSE POP            ( se não puder movimentar apaga o )
   THEN ;              ( incremento deixado na pilha de retorno )
```

Usamos a pilha de retorno para passar o parâmetro incremento de posição (que estabelece o sentido de deslocamento da BASE) pois a pilha já está tomada de parâmetros, sendo mais simples o nosso manuseio ao colocarmos na pilha de retorno.

A palavra a seguir simplesmente testa se foi teclado o comando correspondente à movimentação e se está na faixa permitida de mobilidade:

```
: MOV.BASE
   -1 PUSH             ( coloca incremento negativo na pilha )
                       ( de retorno )
   DUP 136 =           ( verifica se a tecla pressionada é <— )
   XBASE 0 > AND       ( e se a BASE ainda não atingiu o )
   POSIC.BASE          ( canto esquerdo; posiciona a BASE )
   1 PUSH              ( coloca incremento positivo na pilha )
                       ( de retorno )
   DUP 149 =           ( verifica se a tecla pressionada é —> )
   XBASE 29 < AND      ( e se a BASE não atingiu o canto )
   POSIC.BASE          ( direito; posiciona a BASE )
   DELAY 0 DO LOOP ;   ( laço vazio de espera )
```

Abaixo descrevemos a definição da palavra de disparo de TIRO:

```
: NOVO.TIRO
  DUP 224 =                ( verifica se é a tecla (espaço) )
  IF 0 BEGIN               ( repete )
       DUP TIROS PEEK      ( enquanto TIROS estiver ativo )
     WHILE
       1 +                 ( experimenta o próximo )
       10 =                ( se for o último )
       IF DROP             ( então apaga o contador )
         0 9 TIROS         ( desativa o tiro )
         POKE 9            ( e deixa o índice na pilha )
       THEN
     REPEAT
    DUP DUP                ( duplica o índice duas vezes )
    XBASE 1 + SWAP         ( XBASE+1 é armazenado )
    XT POKE                ( no XT do TIRO dado pelo índice )
    YBASE 1 - SWAP         ( YBASE-1 é armazenado )
    YT POKE                ( no YT do TIRO dado pelo índice )
    1 SWAP TIROS POKE      ( ativa este TIRO que estava )
  THEN ;                   ( desativado )
```

Definiremos agora as palavras que são capazes de verificar se algum OVNI foi atingido, além disso, atualizar o flag VENC caso todos os OVNIs tenham sido destruídos:

```
: TEST.FIM
   0                       ( inicializa com zero o topo da )
```

```
  15 0 DO                  ( pilha; repete p/ todos os OVNIs )
       I ATIVOS PEEK       ( soma o valor do flag (1 ou 0) )
       +                   ( p/ todos os OVNIs, caso o
  LOOP                     ( resultado seja 0, nenhum OVNI )
  0 = -> VENC ;            ( está ativo e VENC assume 1 )

: ACER.OVNI
  10 0 DO                  ( para todos os TIROS faça )
   I TIROS PEEK IF         ( verifica se está ativo )
    15 0 DO                ( para todos os OVNIs faça )
     I ATIVOS PEEK IF      ( verifica se está ativo )
       I YP PEEK           ( deixe na pilha YP )
       J YT PEEK           ( deixe na pilha YT )
       2 PICK 1 +          ( verifique se YP+1 )
       2 PICK >=           ( e >= YT )
       PUSH <= PULL        ( e se YP <= YT )
       AND
       I XP PEEK           ( deixa na pilha XP )
       J XT PEEK           ( deixa na pilha XT )
       2 PICK 2 +          ( verifique se XP+2 )
       2 PICK >=           ( e >= XT )
       PUSH <= PULL        ( e se XP <= XT )
       AND
       AND IF              ( e se tudo isso for verdadeiro )
          0 J TIROS POKE   ( apaga o TIRO )
          0 I ATIVOS POKE  ( apaga o OVNI )
          I XP PEEK ->     ( XEXP recebe a coordenada X do )
          XEXP             ( OVNI e )
          I YP PEEK        ( YEXP recebe a coordenada Y do )
          -> YEXP          ( OVNI )
          N.OVNI EXPLODE   ( explode o OVNI atingido )
          I ACRES.PTOS     ( pontos ganhos em função )
       THEN                ( da posição do OVNI )
      THEN
     LOOP
    THEN
   LOOP
   TEST. FIM ;             ( verifica se acabaram os OVNIs )
```

Usaremos uma palavra auxiliar para chamar um conjunto de funções que são chamadas várias vezes, não tendo portanto que reescrever as palavras mais de uma vez:

```
: AUX
    1 1 BLKSIZE
    -936 CALL
```



```
    MOV.TIROS
    GETKEY CLRKEY
    3 2 BLKSIZE
    MOV.BASE
    NOVO.TIRO
    ACER.OVNI DROP ;
```

Já possuímos todos os módulos necessários para a definição da palavra de movimentação dos objetos, como segue:

```
: MOV.OBJ
   3 2 BLKSIZE             ( estabelece tamanho do bloco )
   BEGIN
    -2 1 DO                ( alternando sentido +1 e -1 )
     9 0 DO                ( ande 9 posições nessa direção )
       3 0 DO              ( repete p/ os 3 grupos de OVNIs )
         0 K 5 I 1 + *
         5 I * MOV.OVNI    ( move OVNIs 5*I até 5*(I+1) )
         AUX               ( executo várias rotinas )
       LOOP
     LOOP
    DELAY 10 - -> DELAY    ( laço de espera )
    1 0 15 0 MOV.OVNI      ( move OVNIs 1 linha p/ baixo )
    AUX                    ( executo várias rotinas )
   -2 +LOOP
  DEST VENC OR UNTIL ;     ( testo fim de jogo )
```

Uma vez tendo nosso programa com todas as suas variáveis prontas, podemos definir a palavra de inicialização de variáveis que é responsável simplesmente por colocar nas variáveis strings de coordenadas dos OVNIs os valores das posições iniciais dos mesmos, quando do início do jogo. Além disso, devem ser ativados todos os OVNIs (colocando o valor 1 em ATIVOS), e desativados todos os tiros (colocando o valor 0 em TIROS). Segue a definição:

```
: POS.INI.OVNI
    10  0  DO
        0  I  TIROS POKE       ( zera todos os TIROs )
    LOOP
    15  0  DO
        1  I  ATIVOS POKE      ( coloca todos os OVNIs ativos )
    LOOP
    3  0  DO
      5  0 DO
          15 J 4 * -           ( calcula a posição inicial de )
```

```
            I J 5 * + YP POKE      ( cada OVNI )
        LOOP
    LOOP
    5  0  DO
        3  0  DO
            21 J 5 * -
            J I 5 * + XP POKE
        LOOP
    LOOP ;
```

O programa está pronto, vamos apenas colocar uma apresentação do jogo juntamente com a palavra que executa o programa:

```
: ABERTURA
   ERASE 0 VTAB                    ( apaga a tela e posiciona na )
   PRINT " . . .ATENCAO. . . "     ( 1a. linha, imprimindo a )
   CR 132 PUTC                     ( mensagem e carregando o )
   PRINT " BLOAD CHR.BYTE,A2816 "          ( fonte BYTE )
   CR 3 COLOR 4 CHRSIZE            ( cor 3 e tamanho 4 p/ carac. )
   2816 CHRADR 1 VTAB              ( endereço dos carac.= 2816 )
   PRINT " ATAQUE " CR             ( imprime mensagem nas )
   PRINT " ESPACIAL "              ( linhas 1 e 2 )
   1 CHRSIZE 3 2 BLKSIZE           ( tamanho 1 p/ caracteres )
   CR 132 PUTC                     ( carrega caracteres c/ as )
   PRINT " BLOAD CHR.SPACE,A2816 " CR    ( naves definidas )
   POS.INI.OVNI CHRSET CHRADR ( posiciona inicialmente )
   18 VTAB PRINT " Chegando. . . " ;
: ATAQUE ERASE                 ( apaga a tela )
   0 VTAB 10 HTAB              ( imprime na 1a. linha a mensagem )
   PRINT " PONTOS:000 "
   2816 CHRADR 7 COLOR         ( cor=7 e endereço dos carac.=2816 )
   XBASE HTAB                  ( posiciona a base )
   YBASE VTAB
   13 PUTBLK                   ( imprime o bloco da base )
   BEGIN MOV.OBJ               ( movimenta os objetos )
    DELAY 2 * 3 /              ( reduz o valor de delay 2/3 )
    -> DELAY
    VENC IF                    ( se acabaram os OVNIs )
          PTOS 2 / ACRES.PTOS      ( acrescenta ptos )
          8 VTAB 10 HTAB           ( imprime mensagem )
          PRINT " PRONTO? "
          GETC DROP POS.INI.OVNI  (espera, posiciona OVNIs )
     THEN
   DEST UNTIL                  ( repete até ser destruído )
   5 CHRSIZE 2 HTAB 1 VTAB  ( imprime " FIM " c/ tamanho=5 )
   PRINT " FIM " GETC ;
```


Todo jogo animado deve ser acrescido de alguns efeitos sonoros para torná-lo mais atrativo. Analisaremos como é possível a introdução desse efeito no nosso caso.

A preocupação inicial é onde pode ser colocado esse efeito no nosso programa, para então depois imaginar como isso será feito, e se não for possível essa implementação, então abandonamos essa idéia, partindo para outra mais simples.

As possibilidades mais comuns de inclusão de efeitos sonoros são durante todo o programa, através de uma música contínua durante a partida ou variável conforme a dificuldade, ou localizada em apenas algumas partes, tais como explosões, disparo de tiros ou quando se passa para um nível seguinte.

Uma maneira de se inserir uma música de modo que fique continuamente tocando durante a execução do programa, seria colocando-se em uma tabela os valores das notas musicais e durações correspondentes à música em questão, e inserir em várias partes do programa uma (ou várias) chamada de uma palavra que execute a próxima nota da tabela, apontada por um índice que é continuamente incrementado, e alterado para outra parte da música conforme se queira, por exemplo, na passagem para um outro nível do jogo.

Para o nosso programa, no entanto, simplesmente iremos inserir sons na explosão de uma nave, ou no disparo de um novo tiro. O efeito consiste em se conseguir uma nota ou seqüência de notas, em durações apropriadas, que emitam algum som utilizável segundo o gosto do programador. Geralmente, por ser mais simples de programar, utiliza-se uma seqüência contínua de notas, onde se pode utilizar um simples laço na geração das mesmas (sem a necessidade de montar uma tabela para armazenar as notas).

Podemos inserir na palavra EXPLODE, logo após o início do laço de 0 a 5, a seguinte seqüência de sons:

```
I 1 + 10 * 30 NOTE
```

Também quando um tiro for disparado podemos inserir o seguinte efeito sonoro, logo antes do último THEN:

```
25 80 NOTE
```

Queremos deixar bem claro que são sons simples, sem muitas pretensões, servindo exclusivamente de forma didática para ilustrar as possibilidades. Opções mais eficientes nesses termos podem ser obtidas com valores de notas e seqüências mais bem planejadas, além de se usar valor adequado para a palavra VOICE.

25.4 Técnica de overlay: segmentação do programa

Quando desenvolvemos programas extensos é aconselhável subdividi-los em programas menores, ou seja, segmentos que, por definição, consistem em partes de um programa que não necessariamente precisam estar ao mesmo tempo presentes na memória; e em disco, não fazem parte de um mesmo arquivo. Em outras palavras: um programa extenso é dividido em partes (segmentos) que são editadas e gravadas separadamente em arquivos distintos no disquete, e que são executadas uma de cada vez, saindo uma para entrar outra, minimizando a área de memória utilizada para o programa.

Para um funcionamento adequado, no caso do GraFORTH, devemos ter o cuidado de não usar na parte anterior palavras que sejam usadas na atual, já que quando uma parte é introduzida na memória costuma-se sobrepô-la às palavras anteriores não usadas, liberando, assim, a área de memória disponível no caso.

O overlay (que significa a sobreposição de programas numa mesma área de memória) e a segmentação resolvem também um outro problema: a pequena porção de memória disponível para o EDITOR. Assim, podem-se criar fontes maiores dentro do EDITOR, simplesmente dividindo-o em partes menores.

No nosso caso, a única palavra desnecessária dentro do programa, uma vez executado, é ABERTURA; todas as outras possuem uma utilidade durante o andamento do jogo.

Lembrando que a técnica do overlay consiste em chamar a outra parte do programa, através do READ, cuja primeira linha contém uma chamada da palavra FORGET, seguida da palavra a ser eliminada do dicionário. Deste modo, as definições seguintes ocuparão essa mesma região de memória liberada.

Como só apagaremos a ABERTURA, ela deve ser a última palavra definida na primeira parte. Apresentamos a seguir a listagem completa dos dois arquivos em que se divide ATAQUE.ESPACIAL:

PARTE 1 : ATAQUE.ESPACIAL

```
10    0 VARIABLE PTOS
20    10 VARIABLE XBASE
30    22 VARIABLE YBASE
40    80 VARIABLE DELAY
50    0 VARIABLE DEST
60    0 VARIABLE VENC
70    0 VARIABLE N
80    0 VARIABLE XEXP
```



```
90    0 VARIABLE YEXP
100   15 STRING   YP
110   15 STRING   XP
120   15 STRING   ATIVOS
130   10 STRING   XT
140   10 STRING   YT
150   10 STRING   TIROS
160   : ACRES.PTOS CHRSET CHRADR 15 + PTOS + DUP ->
        PTOS
170     0 VTAB 19 HTAB . 2816 CHRADR ;
180   : EXPLODE 5 0 DO I 2 MOD 0 = IF INVERSE ELSE NOR
        MAL
190     THEN -936 CALL XEXP HTAB YEXP VTAB DUP PUTB
        LK
200     150 0 DO LOOP LOOP NORMAL UNBLK DROP ;
210   : EXP.BASE DEST IF XBASE -> XEXP YBASE -> YEXP
220     13 EXPLODE THEN ;
230   : VERIF.BASE 15 0 DO I ATIVOS PEEK IF I YP PEEK 1 +
240     YBASE = DEST OR -> DEST THEN LOOP ;
250   : N.OVNI 1 6 N * + N 0 = -> N ;
260   : MOV.OVNI DO I ATIVOS PEEK IF I YP PEEK VTAB
270     I XP PEEK HTAB UNBLK DUP I XP PEEK + DUP I XP
        POKE
280     HTAB 2 PICK I YP PEEK + DUP I YP POKE VTAB
290     N.OVNI PUTBLK THEN DELAY 0 DO LOOP LOOP DR
        OP DROP
300     VERIF.BASE EXP.BASE ;
310   : POS.INI.OVNI 10 0 DO 0 I TIROS POKE LOOP 15 0 DO
320     1 I ATIVOS POKE LOOP 3 0 DO 5 0 DO 15 J 4 * -
330     I J 5 * + YP POKE LOOP LOOP 5 0 DO 3 0 DO 21 J 5 *
        -
340     J I 5 * + XP POKE LOOP LOOP ;
350   : ABERTURA ERASE 0 VTAB PRINT " . . .ATENCAO. . .
        "
360     CR 132 PUTC PRINT " BLOAD CHR.BYTE,A2816 " CR
        3 COLOR
370     4 CHRSIZE 2816 CHRADR 1 VTAB PRINT " ATAQUE '
        ' CR
380     PRINT " ESPACIAL " 1 CHRSIZE 3 2 BLKSIZE
390     CR 132 PUTC PRINT " BLOAD CHR.SPACE,A2816 " C
        R
400     POS.INI.OVNI 18 VTAB CHRSET CHRADR
410     PRINT " Chegando. . . " READ " AT.ESP2 " ;
420   CLOSE RUN
```

PARTE 2: AT.ESP2

```
10    FORGET ABERTURA
20    : POSIC.BASE IF XBASE DUP HTAB YBASE VTAB UNB
        LK
30      PULL + DUP -> XBASE HTAB 13 PUTBLK ELSE POP
        THEN ;
40    : MOV.BASE -1 PUSH DUP 136 = XBASE 0 > AND POSIC
        .BASE
50      1 PUSH DUP 149 = XBASE 29 < AND POSIC.BASE
60      DELAY 0 DO LOOP ;
70    : NOVO.TIRO DUP 224 = IF 0 BEGIN DUP TIROS PEEK
        WHILE
80      1 + 10 = IF DROP 0 9 TIROS POKE 9 THEN REPEAT
90      DUP DUP XBASE 1 + SWAP XT POKE YBASE 1 - SWAP
        YT POKE
100     1 SWAP TIROS POKE THEN ;
110   : TEST.FIM 0 15 0 DO I ATIVOS PEEK + LOOP 0 = -> V
        ENC ;
120   : ACER.OVNI 10 0 DO I TIROS PEEK IF 15 0 DO
130     I ATIVOS PEEK IF I YP PEEK J YT PEEK 2 PICK 1 +
140     2 PICK >= PUSH <= PULL AND I XP PEEK J XT PEEK
150     2 PICK 2 + 2 PICK >= PUSH <= PULL AND AND IF
160     0 J TIROS POKE 0 I ATIVOS POKE I XP PEEK -> XEX
        P
170     I YP PEEK -> YEXP N.OVNI EXPLODE I ACRES.PTOS
        THEN
180     THEN LOOP THEN LOOP TEST.FIM ;
190   : MOV.TIROS 10 0 DO I TIROS PEEK IF I YT PEEK 1 -
200     DUP 2 <= IF 0 I TIROS POKE I YT PEEK VTAB
210     I XT PEEK HTAB UNBLK DROP ELSE DUP I YT POKE
220     I XT PEEK 2 PICK 1 + VTAB DUP HTAB UNBLK HTAB
        VTAB
230     21 PUTBLK THEN THEN DELAY 2 / 0 DO LOOP LOOP ;
240   : AUX 1 1 BLKSIZE -936 CALL MOV.TIROS GETKEY C
        LRKEY
250     3 2 BLKSIZE MOV.BASE NOVO.TIRO ACER.OVNI DR
        OP ;
260   : MOV.OBJ 3 2 BLKSIZE BEGIN -2 1 DO 9 0 DO 3 0 DO
270     0 K 5 I 1 + * 5 I * MOV.OVNI AUX LOOP LOOP
280     DELAY 10 - -> DELAY 1 0 15 0 MOV.OVNI AUX
290     -2 +LOOP DEST VENC OR UNTIL ;
300   : ATAQUE ERASE 0 VTAB 10 HTAB PRINT " PONTOS:
        000 "
```



```
310     2816 CHRADR 7 COLOR XBASE HTAB YBASE VTAB
        13 PUTBLK
320     BEGIN MOV.OBJ DELAY 2 * 3 / -> DELAY VENC IF
330     PTOS 2 / ACRES.PTOS 8 VTAB 10 HTAB PRINT " PRO
        NTO? "
340     GETC DROP POS.INI.OVNI THEN DEST UNTIL 5 CHR
        SIZE
350     2 HTAB 1 VTAB PRINT " FIM " GETC
360     PAD ASSIGN " FORGET PTOS ABORT " PAD MEMRD ;
370   CLOSE RUN
```

Não devemos nos esquecer de salvar cada um dos arquivos separadamente, com os nomes atribuídos, ATAQUE.ESPACIAL e AT. ESP2, respectivamente.

Na linha 360, usamos um dos recursos existentes no GraFORTH, ou seja, a capacidade que o programa tem de apagar-se a si mesmo, através do uso da palavra MEMRD, a qual lê uma seqüência de caracteres armazenados em alguma região da memória e entende como chamada de palavras do GraFORTH, executando-as da mesma maneira que se fossem introduzidas via teclado. Desta forma, vamos apagar a palavra PTOS, a primeira definida no programa, e reinicializando o sistema através de ABORT. PAD MEMRD executa os comandos armazenados temporariamente em PAD.

Podemos então inicializar o jogo, bastando para tanto chamar do disco o arquivo onde ele foi armazenado que ele será executado automaticamente, e também automaticamente apagado da memória ao término da partida, sendo necessário reiniciar o jogo caso se queira outra partida. Para tanto basta teclarmos:

```
Ready  READ " ATAQUE.ESPACIAL "
```

e o programa será carregado e executado.

26

DESENVOLVENDO UM PROGRAMA APLICATIVO

26.0 Gráfico de barras

Neste capítulo, enfatizaremos a animação gráfica, ou seja, recursos e técnicas, visando ao desenvolvimento de jogos ou animações do tipo simulação. Porém, acima dessa primeira impressão que se tem, nosso principal objetivo é mostrar várias das possibilidades gráficas do sistema.

O GraFORTH, como linguagem de desenvolvimento que é, apresenta todas as condições e recursos para a elaboração de programas aplicativos e utilitários, mais sérios e de uso possivelmente semiprofissional. O objetivo do programa que desenvolveremos a seguir é mostrar a possibilidade de utilização de recursos gráficos em programas diferentes dos joguinhos.

O programa escolhido será utilizado para a geração de gráficos comparativos em forma de barras, muito utilizados para visualizar dados estatísticos.

Basicamente, o programa deverá mostrar uma tela onde se encontre o gráfico de barras produzido e os dados necessários à sua compreensão, algo semelhante ao apresentado abaixo, onde podemos notar:

- título do gráfico;
- legenda indicando as barras para mais de uma barra sobreposta para um mesmo valor de ordenada (mais de um nível);
- o eixo vertical, com sua escala numérica desde um valor máximo até um mínimo, ambos especificados;
- o eixo horizontal, com as legendas correspondentes a cada uma das barras;
- as barras formando retângulos proporcionais ao valor que eles representam, sendo que um mesmo nível apresenta uma mesma cor.

Para se reproduzir esse gráfico, o programa deve ler os dados de entrada a serem utilizados, o que pode ser feito de várias formas: introdução através de tabela, ou um arquivo de dados, ou ainda introdução diretamente, ou seja, introduzir o dado e desenhar a barra antes de ler o próximo.

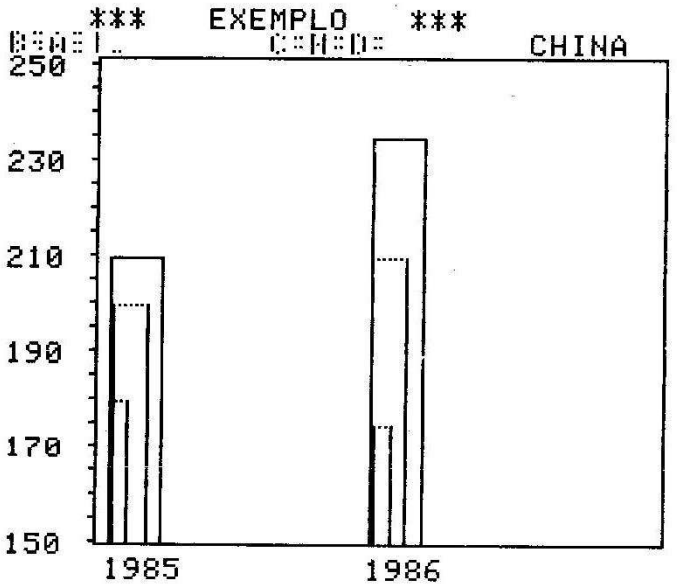


Figura 26.0 – A tela do GRÁFICO DE BARRAS

No nosso caso, montaremos inicialmente uma tabela com os valores correspondentes de cada barra, em cada nível. Além disso, deve-se também ler o título do gráfico, o número de barras (limitado entre 2 e 9), de onde se pode saber o número de caracteres máximo para cada um dos nomes das barras (legenda no eixo horizontal), número de níveis de barras possíveis sob uma mesma ordenada (1 a 6), sendo um para cada cor e uma legenda para os níveis, caso o número de níveis seja maior do que 1.

A seguir apresentamos o DHF correspondente ao programa, que basicamente é dividido em duas partes, uma onde os dados são lidos e outra onde é produzido o gráfico propriamente dito.

Por uma questão de limite de algarismos significativos nas operações realizadas com o GraFORTH, só usaremos valores de um byte no nosso programa, visto que em operações de multiplicação com números de dois bytes pode ocorrer 'overflow' (o valor ultrapassar o limite armazenável pelo sistema), tornando os resultados obtidos totalmente errôneos. No mais, a única rotina um pouco mais complexa é a que realmente tem de calcular as dimensões das barras, que correspondem ao núcleo básico do programa.

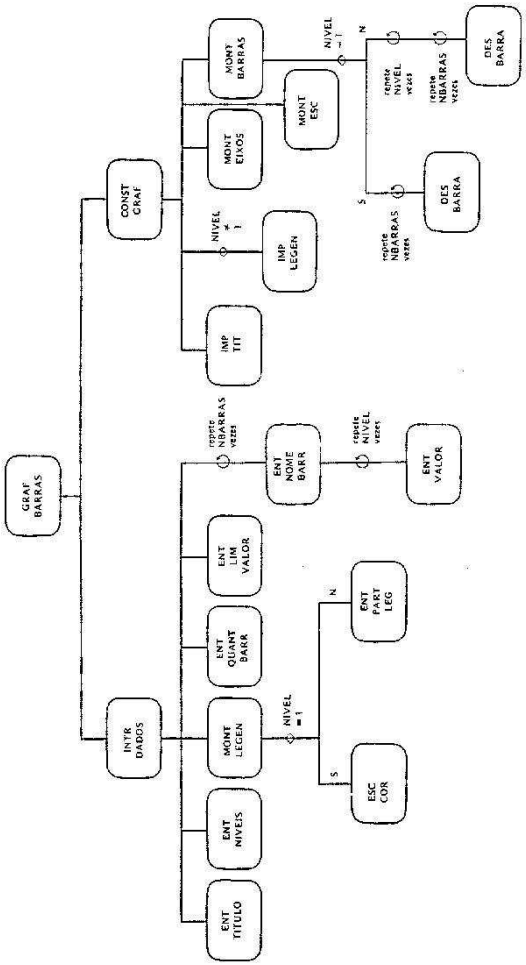


Figura 26.1 – DHF do GRÁFICO DE BARRAS

26.1 As duas partes: entrando os dados e desenhando os gráficos

Usaremos a técnica de segmentação descrita anteriormente neste programa, uma vez que ele é propício a essa separação, visto que apresenta duas partes nítidas: a introdução dos dados e a produção do gráfico. Uma vez usando dois segmentos para o programa, conseguimos sua utilização mais racional da memória, já que apenas as variáveis que contêm os dados permanecerão na memória, enquanto que as palavras que foram usadas para introduzir esses valores são apagadas.

A seguir, apresentamos a listagem comentada das duas partes, no formato adequado à introdução no editor, sendo apenas desnecessários os comentários entre parênteses. Sua utilidade é exclusivamente auxiliar na compreensão do programa.

26.1.0 Primeira parte: GRAF.BARRAS

```
10   55 STRING VALOR                    ( definição das variáveis )
20   45 STRING $TITULO                  ( utilizadas para armazenar os )
30   1 VARIABLE NIVEL                   ( dados introduzidos )
40   0 VARIABLE C/LEG
50   45 STRING $LEGEN
60   0 VARIABLE COR
70   0 VARIABLE NBARRAS
80   0 VARIABLE NC/E
90   40 STRING $NOMBAR
100  0 VARIABLE VMIN
110  0 VARIABLE VMAX
120  : LIMP.STR 55 0 DO 0 I VALOR POKE      ( limpa strings )
130    LOOP 45 0 DO 224 I $TITULO POKE      ( inicial. c/ bran- )
140    224 I $LEGEN POKE LOOP               ( cos ou zeros )
150  40 0 DO 224 I $NOMBAR POKE LOOP ;
160  : INTERV 3 PICK >= PUSH >= PULL   ( testa se um valor )
170    AND ;                           ( pertence a um intervalo )
140  : ENT.TITULO CR PRINT                ( entra título do )
150    " TITULO DO GRAFICO (1 a 40) "     ( gráfico )
160    CR PRINT " => " 0 $TITULO
170    READLN ;
180  : ENT.NIVEIS BEGIN CR PRINT          ( entra níveis )
190    " NIVEIS DE BARRAS (1 a 6): "
200    GETC 176 – DUP . –> NIVEL   ( obtém o código e armaz. )
210    NIVEL 1 6 INTERV ULTIL ( em NIVEL, se for válido calc. )
220    40 NIVEL / 2 – –> C/LEG ;   ( núm. de carac. permitidos )
230  : ENT.PART.LEG CER PRINT       ( lê as legendas para os )
```

```
240   " LEGENDA PARA OS NIVEIS " CR                ( níveis )
250   PRINT " MAX DE " C/LEG . SPCE
260   PRINT " CARACT. P/ NIVEL "
270   NIVEL 0 DO CR I . SPCE PRINT
280   " => "I C/LEG 2 + * 1 + $LEGEN
290   READLN LOOP ;
300 : ESC.COR BEGIN CR PRINT   ( entra com a cor p/ o gráf. )
310   " ESCOLHA COR P/ BARRAS: "
320   GETC 176 - DUP DUP . -> COR
330   1 7 INTERV UNTIL ;
340 : MONT.LEGEN NIVEL 1 > IF      ( verifica valor do nível )
350   ENT.PART.LEG ELSE DROP
360   ESC.COR THEN ;
370 : ENT.QUANT.BAR BEGIN CR PRINT  ( entra qtde. barras )
380   " QTAS BARRAS? (2 a 9): " GETC
390   176 - DUP . DUP -> NBARRAS
400   2 9 INTERV UNTIL                ( calcula núm. de caract. )
410   27 NBARRAS / 1 - -> NC/E ;        ( p/ nome das barras )
420 : ENT.NOME.BARRA CR PRINT   ( entra nome das barras )
430   " NOME DO ELEMENTO " I 1 + .
450   SPCE PRINT " (2 a " NC/E .
460   PRINT " ) " CR PRINT " => "
470   I NC/E 1 + * $NOMBAR READLN ;
480 : ENT.VALOR CR PRINT " NIVEL "   ( entra os valores p/ )
490   I . SPCE PRINT " VALOR: " PAD              ( cada nível )
500   READLN PAD GETNUM
510   I 9 * J + VALOR POKE ;
520 : ENT.LIM.VALOR CR PRINT   ( entra valores máx. e mín. )
530   " ENTRE VALOR MINIMO: "
540   PAD READLN PAD GETNUM -> VMIN
550   CR PRINT " ENTRE VALOR MAXIMO: "
560   PAD READLN PAD GETNUM -> VMAX ;
570 : ENT.ELEM NBARRAS 0 DO    ( entra c/ os dados p/ cada )
580   ENT.NOME.BARRA NIVEL 0 DO      ( barra e cada nível )
590   ENT.VALOR LOOP LOOP ;
600 : INT.DADOS LIMP.STR ENT.TITULO       ( rotina básica )
610   ENT.NIVEIS MONT.LEGEN
620   ENT.QUANT.BAR ENT.LIM.VALOR
630   ENT.ELEM READ " GER.BARRAS " ;
640 CLOSE RUN
```

26.1.1 Segunda parte: GER.BARRAS

```
10  FORGET LIMP.STR
20  0 VARIABLE X                                ( cria variáveis )
30  0 VARIABLE Y
```



```
40  0 VARIABLE LX
50  0 VARIABLE LY
60  : COLORTAB                          ( cria tabela de cores )
70    1 , 2 , 3 , 5 , 6 , 7 , ;
80  : RETANGULO -> Y -> X 128 PEEK  ( desenha retângulo )
90    -> LX 130 PEEK -> LY
100   LX Y LINE X Y LINE X LY LINE
110   LX LY LINE ;
120 : IMP.TIT ERASE 0 VTAB 0 HTAB( imprime o título na 1ª )
130   0 $TITULO WRITELN CR ;                        ( linha )
140 : PRTCOLOR 1 CHRSIZE I ' COLORTAB( obtém cor c/ que )
150   + PEEK COLOR ;              ( as palavras serão exibidas )
160 : IMP.LEGEN NIVEL 1 <> IF   ( imprime as legendas dos )
170   NIVEL 0 DO PRTCOLOR I C/LEG 2 +              ( níveis )
180   * DUP HTAB 1 VTAB 1 + $LEGEN
190   WRITELN LOOP 0 CHRSIZE THEN ;
200 : MONT.EIXOS 32 16 POSN          ( desenha os eixos do )
210   230 179 RETANGULO 183 18 DO                ( gráfico )
220   29 I POSN 32 I LINE 8 +LOOP ;
230 : MONT.ESC 23 2 DO 0 HTAB I VTAB  ( imprime escala no )
240   22 I - VMAX VMIN * 20 /                ( eixo vertical e )
250   VMIN + . 4 +LOOP CR          ( os nomes das barras no )
260   NBARRAS 0 DO I NC/E 1 + * DUP       ( eixo horizontal )
270   5 + HTAB $NOMBAR WRITELN LOOP ;
280 : DES.BARRA NBARRAS 0 DO       ( para um dado nível )
290   COR COLOR I NC/E 1 + * 5 +          ( desenha as barras )
300   7 * 2 + DUP J 9 * I + VALOR PEEK
310   VMIN - -161 * VMAX VMIN - /
320   179 + POSN J 1 + 18 * NIVEL /
330   + 179 RETANGULO LOOP ;
340 : MONT.BAR NIVEL 1 = IF 1 0 DO   ( desenha as barras p/ )
350   DES.BARRA LOOP ELSE              ( um ou mais níveis )
360   NIVEL 0 DO I ' COLORTAB + PEEK
370   -> COR DES.BARRA LOOP THEN ;
380 : CONST.GRAF IMP.TIT IMP.LEGEN    ( palavra principal )
390   MONT.EIXOS MONT.ESC MONT.BAR
400   23 VTAB 0 HTAB GETC ABORT ;
410 CLOSE RUN
```

26.2 Um exemplo de aplicação para GRÁFICO DE BARRAS

Em muitas situações, a representação de dados através de relatórios não fornece uma visão comparativa de conjunto. Gráficos, desenhos e figuras são bem mais significativos. O nosso aplicativo serve para este fim. Iremos aqui mostrar um exemplo de utilização desse GERADOR DE GRÁFICOS.

Imaginemos uma família que deseja analisar o andamento de seu orçamento doméstico nos últimos anos e tenha montado a seguinte tabela:

DESPESAS FAMILIARES (*10 Cz$)

ANOS	VESTUÁRIO	MORADIA	TRANSPORTE	ALIMENTAÇÃO
1982	20	30	50	60
1983	30	50	80	90
1984	60	70	110	120
1985	110	180	120	170
1986	120	200	140	210

Os valores na tabela representam o correspondente a um décimo do valor em cruzados (ou seja, 20 na tabela representa Cz$200). Isto é feito para que os valores tabelados se situem entre 0 e 255, valores adequados para os cálculos. Cada coluna da tabela corresponderá a um nível do nosso programa, no caso, 4. Teremos uma barra para cada ano, sendo 5 no total.

Notemos os valores máximos e mínimos na tabela: encontramos 20 para o valor mínimo e 210 para o máximo. Porém, expandiremos um pouco os limites: para o valor mínimo usaremos 0 e para o máximo o valor 230.

A utilização do programa é simples. Inicialmente chamamos o programa da memória e ele faz todas as perguntas necessárias. Deve-se tomar cuidado com a entrada de dados, já que o programa não faz nenhuma consistência; a entrada de um valor incorreto causa a reinicialização do programa (⟨CTRL⟩⟨RESET⟩ seguido de RUN) para uma nova entrada de dados.

```
Ready  READ " GRAF.BARRAS "
Ready  RUN

TITULO DO GRAFICO (1 A 40)
=> DESPESAS FAMILIARES (*10 Cz$)

NIVEIS DE BARRAS (1 A 6)
=>4

LEGENDA DE CORES PARA OS NIVEIS
MAX DE 8 CARAC. P/ NIVEL

0 =>VESTUAR
1 =>MORADIA
```



```
2 =>TRANSP
3 =>ALIMENT

QTAS BARRAS (2 A 9): 5
ENTRE VALOR MINIMO: 0
ENTRE VALOR MAXIMO: 230

NOME DO ELEMENTO 1 (2 A 4)
=>1982
NIVEL 0 VALOR: 20
NIVEL 1 VALOR: 30
NIVEL 2 VALOR: 50
NIVEL 3 VALOR: 60

NOME DO ELEMENTO 2 (2 A 4)
=>1983
NIVEL 0 VALOR: 30
NIVEL 1 VALOR: 50
NIVEL 2 VALOR: 80
NIVEL 3 VALOR: 90

NOME DO ELEMENTO 3 (2 A 4)
=>1984
NIVEL 0 VALOR: 60
NIVEL 1 VALOR: 70
NIVEL 2 VALOR: 110
NIVEL 3 VALOR: 120

NOME DO ELEMENTO 4 (2 A 4)
=>1985
NIVEL 0 VALOR: 110
NIVEL 1 VALOR: 180
NIVEL 2 VALOR: 120
NIVEL 3 VALOR: 170

NOME DO ELEMENTO 5 (2 A 4)
=>1986
NIVEL 0 VALOR: 120
NIVEL 1 VALOR: 200
NIVEL 2 VALOR: 140
NIVEL 3 VALOR: 210
```

Terminada a introdução dos dados é gerado o gráfico de barras correspondentes. Note que aparece no canto esquerdo inferior um cursor piscante, esperando que qualquer tecla seja pressionada para finalizar o programa.

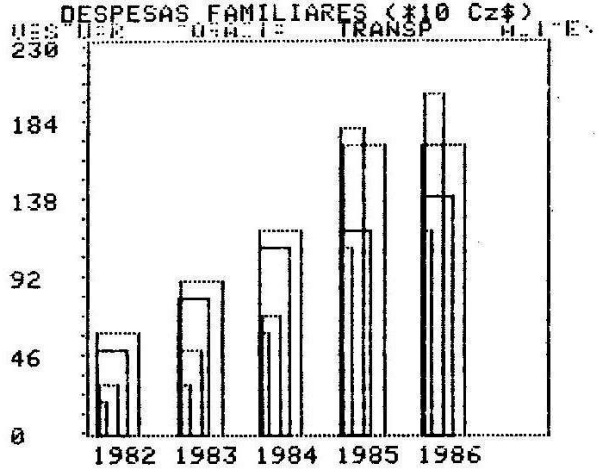


Figura 26.2 – Exemplo de Graf.BARRAS.

27

ELABORANDO UMA ANIMAÇÃO TRIDIMENSIONAL

27.0 Animação tridimensional na apresentação do sistema

Após termos apresentado exemplos dos recursos do sistema quanto a caracteres gráficos e gráficos bidimensionais, mostraremos também exemplos de desenvolvimento de programas, usando os recursos tridimensionais do sistema GraFORTH. Para tanto, usaremos como exemplo uma pequena animação que servirá como uma apresentação do sistema, executada sempre que ele for carregado. Esta animação consiste basicamente em escrever a palavra "GraFORTH" com letras tridimensionais que surgem e se movimentam na tela das mais variadas formas, de modo a utilizar o máximo dos recursos disponíveis, aproveitando a oportunidade para exemplificar a utilização de alguns editores tridimensionais, tais como o IMAGEDITOR.

Como primeira etapa no desenvolvimento deste e de outros programas que utilizem objetos tridimensionais, temos a definição destes objetos numa forma inteligível e utilizável pelo sistema, que consiste na criação do arquivo de imagem do objeto via IMAGEDITOR.

Uma vez montado o arquivo de imagem do objeto, devemos estabelecer valores para os atributos do mesmo. Ou seja, continuamente redesenhando o objeto com novos fatores de escala, novos ângulos de rotação e novas posições na tela, de forma a dar uma noção de movimento.

Esta seqüência deverá surgir de maneira contínua, ou através de dados previamente tabelados ou de uma função matemática que represente a sucessão contínua de estados em que o objeto se encontrará. Se utilizarmos uma tabela que contém os dados já calculados, poderemos ganhar em velocidade, já que ganhamos o tempo gasto com o cálculo de novos valores da função a cada intervalo entre dois quadros sucessivos. No nosso caso, porém, usaremos funções que calculam o valor a cada etapa da animação.

O mais importante no momento, entretanto, é justamente definir uma função que se ajuste ao movimento que nós queremos. No nosso exemplo, usaremos basicamente funções lineares, que correspondem a obter valores igualmente distribuídos dentro de um inter-

valo. Por exemplo, se desejássemos obter 5 valores entre 3 e 74, usaríamos o seguinte laço em GraFORTH:

```
6 0 DO
    74 3 - I * 5 /
LOOP
```

com o qual seriam armazenados na pilha, em ordem de tempo de execução, os números: 0 14 28 42 56 71.

Este processo simples de geração dos valores intermediários para os atributos dos objetos será usado para obtenção de praticamente todos os valores que necessitaremos, embora outras funções pudessem ser utilizadas.

27.1 Usando o IMAGEDITOR: Criação de cada letra

Para iniciar a montagem do programa de apresentação do sistema, devemos definir as letras da palavra GraFORTH como objetos tridimensionais, manipuláveis pelo sistema, de forma a poder exibi-las posteriormente na tela, nas diversas posições que estipularemos mais tarde. Para isso, faremos uso do IMAGEDITOR, que é utilizado justamente para a criação de imagens tridimensionais.

Para carregar o IMAGEDITOR, devemos primeiramente esquecer (FORGET) as palavras novas que você tinha inserido no dicionário e, em seguida, utilizar os comandos:

```
Ready ABORT
Ready READ " IMAGEDITOR "
Ready RUN
```

Definiremos cada letra como um objeto, e, desta forma, cada letra deve possuir seu próprio arquivo de imagem.

A seguir apresentamos os passos a serem seguidos na criação de uma letra, que devem ser repetidos para todas as letras e, em seguida, os valores numéricos a serem introduzidos para cada letra, com seu respectivo nome de arquivo.

1) comando: A – entre com o endereço do objeto igual a 2816;

2) comando: Z – apague o arquivo de imagem que porventura já esteja ali;

3) comando: W – apague a tela e estabeleça linha 16 para o topo da janela de texto;

4) comando: E – entre com os dados do arquivo de imagem que está sendo editado no momento. Siga os valores apresentados na listagem a seguir para cada letra. Utilize a tecla (←) caso algum dado tenha sido introduzido erroneamente. Note, porém, que essa tecla só

pode ser usada depois de uma linha de dados ter sido introduzida. Isso quer dizer que se você errou o valor da coordenada x, deverá introduzir quaisquer valores nas coordenadas y e z, para depois apagar toda essa linha e reintroduzi-la. Repita para todos os dados de um arquivo;

5) comando: K – grave o arquivo de imagem recém-concluído com o respectivo nome;

6) repita desde o passo 1 até que não existam mais objetos (letras) a serem definidos;

7) comando: Q – abandone o editor.

Figura 27.0 – Listagem de definição da palavra "GraFORTH"

Definição da letra G

```
2816  M  0    0    100
2820  D  100   0    100
2824  D  100   100  100
2828  D  -100  100  100
2832  D  -100  -100 100
2836  D  100   -100 100
2840  D  100   -100 -100
2844  D  -100  -100 -100
2848  D  -100  100  -100
2852  D  100   100  -100
2856  D  100   0    -100
2860  D  0     0    -100
2864  D  0     0    100
2868  M  100   0    100
2872  D  100   0    -100
2876  M  100   100  100
2880  D  100   100  -100
2884  M  -100  100  100
2888  D  -100  100  -100
2892  M  -100  -100 100
2896  D  -100  -100 -100
```

Definição da letra R

```
2816  M  0    0    100
2820  D  -100  20   100
2824  M  -100  0    100
```

```
2828  D  -100   100   100
2832  D  -100   100  -100
2836  D  -100     0  -100
2840  D  -100     0   100
2844  M  -100    20  -100
2848  D     0     0  -100
2852  D     0     0   100
```

Definição da letra A

```
2816  M  -100     0   100
2820  D     0     0   100
2824  D     0   100   100
2828  D  -100   100   100
2832  D  -100    50   100
2836  D     0    50   100
2840  M  -100     0  -100
2844  D     0     0  -100
2848  D     0   100  -100
2852  D  -100   100  -100
2856  D  -100    50  -100
2860  D     0    50  -100
2864  M     0     0  -100
2868  D     0     0   100
2872  M  -100     0  -100
2876  D  -100     0   100
2880  M  -100    50  -100
2884  D  -100    50   100
2888  M     0   100  -100
2892  D     0   100   100
2896  M  -100   100  -100
2900  D  -100   100   100
```

Definição da letra F

```
2816  M   100  -100   100
2820  D  -100  -100   100
2824  D  -100   100   100
2828  D  -100   100  -100
2832  D  -100  -100  -100
2836  D   100  -100  -100
```



```
2840  D   100  -100   100
2844  M  -100  -100   100
2848  D  -100  -100  -100
2852  M  -100     0   100
2856  D     0     0   100
2860  D     0     0  -100
2864  D  -100     0  -100
```

Definição da letra O

```
2816  M   100  -100   100
2820  D  -100  -100   100
2824  D  -100   100   100
2828  D   100   100   100
2832  D   100  -100   100
2836  D   100  -100  -100
2840  D  -100  -100  -100
2844  D  -100   100  -100
2848  D   100   100  -100
2852  D   100  -100  -100
2856  M  -100  -100   100
2860  D  -100  -100  -100
2864  M  -100   100   100
2868  D  -100   100  -100
2872  M   100   100   100
2876  D   100   100  -100
```

Definição da letra R

```
2816  M  -100   100   100
2820  D  -100  -100   100
2824  D   100  -100   100
2828  D   100     0   100
2832  D  -100     0   100
2836  D   100   100   100
2840  D   100   100  -100
2844  D  -100     0  -100
2848  D   100     0  -100
2852  D   100  -100  -100
2856  D  -100  -100  -100
2860  D  -100   100  -100
```

```
2864  D  -100   100   100
2868  M  -100  -100   100
2872  D  -100  -100  -100
2876  M   100  -100  -100
2880  D   100  -100   100
2884  M   100     0  -100
2888  D   100     0   100
```

Definição da letra T

```
2816  M  -100  -100   100
2820  D   100  -100   100
2824  D   100  -100  -100
2828  D  -100  -100  -100
2832  D  -100  -100   100
2836  M     0  -100   100
2840  D     0   100   100
2844  D     0   100  -100
2848  D     0  -100  -100
```

Definição da letra H

```
2816  M  -100  -100   100
2820  D  -100   100   100
2824  D  -100   100  -100
2828  D  -100  -100  -100
2832  D  -100  -100   100
2836  M   100  -100   100
2840  D   100   100   100
2844  D   100   100  -100
2848  D   100  -100  -100
2852  D   100  -100   100
2856  M  -100     0   100
2860  D   100     0   100
2864  M  -100     0  -100
2868  D   100     0  -100
```


27.2 A montagem do programa

Inicialmente, faremos aparecer a palavra FORTH, com escala vertical (eixo y) crescente de 0 ao valor estipulado, porém, com a escala na horizontal com um valor negativo, de forma que as letras apresentem-se invertidas. Além disso, as letras são mostradas de frente, na horizontal, segundo a figura 27.1.

Figura 27.1 — Apresentação dos caracteres GraFORTH em posição horizontal.

As coordenadas de posição de cada letra dependem única e exclusivamente da orientação onde elas se encontram e do tamanho (escala) por elas ocupado. Note que foi deixado espaço entre as letras 'G' e 'F' para que pudessem ser incluídas as letras 'r' e 'a'.

Faremos as letras minúsculas 'r' e 'a' saírem dos dois cantos superiores ('r' do direito e 'a' do esquerdo), girando em torno dos três eixos, obrigatoriamente cruzando uma sobre a outra e se posicionando nos lugares reservados. Enquanto isso ocorre, as letras que formam a palavra FORTH (excluindo-se a letra G) giram em torno do eixo x até 90 graus. A seguir mostramos as equações responsáveis pelo posicionamento das letras 'r' e 'a' no seu trajeto, desde uma posição inicial (Xi,Yi) até uma final (Xf,Yf), dependendo da letra em questão:

$$Y = I/N * (Yf-Yi) + Yi$$
$$X = I/N * (Xf-Xi) + Xi$$

onde I varia desde 0 até N, o número de etapas da movimentação.

Para finalizar o nosso exemplo, faremos com que a palavra formada (GraFORTH) fique numa posição em diagonal uma em relação às outras (veja a figura 27.2), dando-nos a impressão de perspectiva e volume. Uma vez posicionadas, as letras invertidas são colocadas na posição correta, terminando assim a nossa pequena animação.

A estrutura lógica do programa está montada, ou seja, foram dados os passos que devem ser executados, bastando agora única e exclusivamente desenvolver os módulos encarregados dessas funções. Apresentamos abaixo a definição desses módulos devidamente comentados para melhor compreensão (a listagem definitiva será apresentada no final do capítulo para ser introduzida no editor). Note que os símbolos entre '<' e '>' representam constantes que devem ser inseridas naqueles lugares e que serão calculadas posteriormente.

```
TESTA-FIM:
GETKEY 127 < IF              ( verifica se alguma tecla )
          CLRKEY
          0 AUTORUN          ( foi pressionada e ABORTa, )
          ABORT              ( desligando o modo AUTORUN )
       THEN

POSICIONA-NA-HORIZONTAL:
5 OBJECT
⟨XG⟩ XPOS ⟨YG⟩ YPOS           ( posiciona a letra G )
0 OBJECT
⟨XF⟩ XPOS ⟨YF⟩ YPOS           ( posiciona a letra F )
1 OBJECT
⟨XO⟩ XPOS ⟨YO⟩ YPOS           ( posiciona a letra O )
2 OBJECT
⟨XR⟩ XPOS ⟨YR⟩ YPOS           ( posiciona a letra R )
3 OBJECT
⟨XT⟩ XPOS ⟨YT⟩ YPOS           ( posiciona a letra T)
4 OBJECT
⟨XH⟩ XPOS ⟨YH⟩ YPOS           ( posiciona a letra H )
⟨escala⟩ 0 DO                ( variando a escala )
    6 0 DO                   ( executa para os 6 objetos )
          I OBJECT           ( referencie o objeto )
          -⟨escala⟩ SCALX    ( atribua ⟨escala⟩ negativa a X )
          J SCALY            ( novo valor da escala em Y )
    LOOP TESTFIM DRAW        ( desenhe os 6 objs. c/ a nova )
LOOP                         ( escala, verif. se alguma )
                             ( tecla foi pressionada )

TRANSLADA-LETRAS:
⟨n⟩ 0 DO                     ( repete para ⟨n⟩ posições )
    I 64 * ⟨n⟩ / CHS         ( calcula o ângulo em X )
    5 0 DO                   ( para os 5 objetos )
        I OBJECT             ( referencie o objeto )
        DUP XROT             ( gire para o ângulo na pilha )
    LOOP DROP                ( apague o ângulo da pilha )
    I ⟨escala⟩ * ⟨n⟩ /       ( calcula o valor da escala )
    I 256 * ⟨n⟩ /            ( calcula o valor do ângulo )
    8 6 DO                   ( para os objetos 6 e 7 )
        I OBJECT             ( referencie o objeto )
        2 PICK SCALE         ( pegue na pilha a escala )
        DUP DUP              ( pegue o ângulo a ser usado )
        XROT YROT            ( nos eixos X e Y )
    LOOP DROP DROP           ( apague os valores na pilha )
    6 OBJECT                 ( para o objeto 6 )
    ⟨Yrf⟩ ⟨Yri⟩ - I *        ( calcule a nova posição X,Y )
```



```
    ⟨n⟩ / ⟨Yri⟩ + YPOS       ( em uma trajetória retilínea )
    ⟨Xrf⟩ ⟨Xri⟩ - I *        ( que deveremos posicionar o )
    ⟨n⟩ / ⟨Xri⟩ + XPOS       ( centro do objeto )
    7 OBJECT                 ( repita para o objeto 7 )
    ⟨Yaf⟩ ⟨Yai⟩ - I *
    ⟨n⟩ / ⟨Yai⟩ + YPOS
    ⟨Xaf⟩ ⟨Xai⟩ - I *
    ⟨n⟩ / ⟨Xai⟩ + XPOS
    5 OBJECT DRAW            ( referencie 5 e desenhe )
LOOP

POSICIONA EM DIAGONAL:
⟨n⟩ 0 DO                     ( repete para ⟨n⟩ posições )
    8 0 DO                   ( repete para os 8 objetos )
        I OBJECT             ( referencie o objeto )
        I 5 < IF             ( para os objs. de 0 a 4 )
          J 24 *⟨n⟩ /        ( calcula o ângulo de giro )
          -64 + XROT         ( de -64 a -40 em X )
          J 16 *⟨n⟩ /        ( e de 0 a 16 em Z )
          ZROT
        ELSE                 ( para os objs. de 5 a 7 )
        J 24 * ⟨n⟩ / XROT    ( giro de 0 a 24 em X )
        J 16 * ⟨n⟩ / YROT    ( e de 0 a 16 em Y )
        THEN
        I 4 > IF             ( deixa na pilha a posição )
            I 5 -            ( horizontal relativa de )
          ELSE               ( cada letra G=0, r=1, a=2 )
            I 3 +            ( F=3, O=4, R=5, T=6, H=7 )
          THEN
        J * ⟨n⟩⟨phf⟩⟨phi⟩     ( calcula posição vertical )
        - * ⟨n⟩ / 7 / ⟨phi⟩ + ( para onde serão deslocados )
        YPOS                 ( os objetos )
    LOOP DRAW                ( desenha os objetos )
LOOP

REVERTE-LETRAS:
⟨escala⟩ 1 + -⟨escala⟩ DO    ( alterando a escala )
    6 0 DO                   ( para os 6 objetos )
        I OBJECT             ( referencia )
        J SCALX              ( altera a escala )
    LOOP
    6 OBJECT 7 OBJECT        ( referencia 6 e 7 )
    DRAW                     ( desenha-os )
LOOP
```

A listagem acima é praticamente definitiva, faltando, porém, determinar o valor das constantes nela presentes.

Esses valores são atribuídos tendo-se em vista a ocupação destas figuras na tela, e procurando sempre evitar as deformações oriundas de os limites serem ultrapassados.

Inicialmente, vamos atribuir os valores para ⟨phi⟩ e ⟨phf⟩. ⟨phi⟩ corresponde à coordenada y dos centros das letras quando elas estão todas alinhadas, pouco abaixo do centro da tela, onde 130 é um valor adequado. ⟨phf⟩ corresponde à coordenada y do centro da letra posicionada mais acima, quando posicionamos as letras em diagonal na tela, próximo do valor 60.

As letras 'r' e 'a' inicialmente partem de alguma posição em cantos opostos. Podemos atribuir quaisquer valores para essas coordenadas, desde que não sejam muito próximos das extremidades. Desta forma, os valores ⟨xri⟩=230, ⟨Yri⟩=20, ⟨xai⟩=10 e ⟨yai⟩=10 são valores adequados. Para os valores finais, já definimos que ⟨yrf⟩=⟨yaf⟩= 130, bastando determinar ⟨xrf⟩ e ⟨xaf⟩.

Já possuímos as coordenadas y iniciais das letras (considerando-se na horizontal), faltando agora descobrir os valores das coordenadas x. Isso será feito dividindo-se o número de letras pelo número de pontos disponíveis (256), sabendo dessa forma o espaço disponível para cada letra.

No entanto, definimos as letras 'r' e 'a' com metade do tamanho das demais. Desta forma podemos imaginar que possuímos apenas 7 letras, o que resulta em 36 pontos para cada letra, restando 4 pontos, 2 de cada lado, que serão deixados como espaço entre os extremos.

Desta forma temos que o centro da letra 'G' está na metade do comprimento disponível para ela (18) mais 2 de espaço lateral, logo ⟨XG⟩=20. Deixando 2 pontos separando 'G' de 'r', e lembrando que 'r' só ocupa metade do tamanho das outras letras e que seu centro está na extremidade direita, obtemos para a posição do centro 2 + 36 + 2 + 16 = 56, ⟨XRF⟩=56. Utilizando a mesma idéia para as letras seguintes obtemos: ⟨xaf⟩=73, ⟨xf⟩=92, ⟨xo⟩=128, ⟨xr⟩=164, ⟨xt⟩= 200, ⟨xh⟩=235.

A seguir, listamos em sua listagem definitiva a apresentação que acabamos de definir, pronta para ser inserida através do editor:

27.3 Apresentação

```
10 FORGET IT
20 : IT ;
30 : FIM
40 0 AUTORUN FORGET IT ABORT ;
```



```
50 : TESTFIM GETKEY 127 > IF CLRKEY
60 FIM THEN ;
70 : POS.HOR 5 OBJECT 20 XPOS 130 YPOS
80 0 OBJECT 92 XPOS 130 YPOS
90 1 OBJECT 128 XPOS 130 YPOS
100 2 OBJECT 164 XPOS 130 YPOS
110 3 OBJECT 200 XPOS 130 YPOS
120 4 OBJECT 235 XPOS 130 YPOS
130 5 1 + 0 DO 6 0 DO I OBJECT
140 5 CHS SCALX J SCALY LOOP DRAW
150 TESTFIM LOOP ;
160 : TRANS.LETR 14 1 + 0 DO I 64 * 14 /
170 CHS
180 5 0 DO I OBJECT DUP XROT LOOP DROP
190 I 5 * 14 / I 255 * 14 / 8 6 DO
200 I OBJECT 2 PICK SCALE DUP DUP
210 XROT YROT LOOP DROP DROP
220 6 OBJECT 130 20 - I * 14 / 20 +
230 YPOS 55 230 - I * 14 / 230 + XPOS
240 7 OBJECT 130 10 - I * 14 / 10 +
250 YPOS 73 10 - I * 14 / 10 + XPOS
260 5 OBJECT DRAW TESTFIM LOOP ;
270 : CONV.I I 4 > IF I 5 -
280 ELSE I 3 + THEN ;
290 : POS.DIAG 14 1 + 0 DO 8 0 DO
300 I OBJECT I 5 < IF J 20 * 14 / -64 +
310 XROT J 16 * 14 / ZROT ELSE
320 J 20 * 14 / XROT J 16 * 14 / YROT
330 THEN CONV.I J * 60 130 - *
340 14 / 7 / 130 + YPOS LOOP
350 DRAW TESTFIM LOOP ;
360 : REVERTE 5 1 + 5 CHS DO
370 6 0 DO I OBJECT J SCALX LOOP
380 6 OBJECT 7 OBJECT DRAW
390 TESTFIM LOOP ;
400 : APRES ERASE
410 CR 132 PUTC PRINT " BLOAD LETRA.F,A2816 " CR
420 CR 132 PUTC PRINT " BLOAD LETRA.O,A2870 " CR
430 CR 132 PUTC PRINT " BLOAD LETRA.R,A2938 " CR
440 CR 132 PUTC PRINT " BLOAD LETRA.T,A3018 " CR
450 CR 132 PUTC PRINT " BLOAD LETRA.H,A3058 " CR
460 CR 132 PUTC PRINT " BLOAD LETRA.G,A3118 " CR
470 CR 132 PUTC PRINT " BLOAD LETRA.r,A3206 " CR
480 CR 132 PUTC PRINT " BLOAD LETRA.A,A3250 " CR OBJ
                                              ERASE ORMODE
490 0 OBJECT 2816 OBJADR
```

```
500 1 OBJECT 2870 OBJADR
510 2 OBJECT 2938 OBJADR
520 3 OBJECT 3018 OBJADR
530 4 OBJECT 3058 OBJADR
540 5 OBJECT 3118 OBJADR
550 6 OBJECT 3206 OBJADR 0 SCALE
560 7 OBJECT 3250 OBJADR 0 SCALE
570 POS.HOR TRANS.LETR POS.DIAG
580 REVERTE 32000 0 DO TESTFIM LOOP FIM ;
```

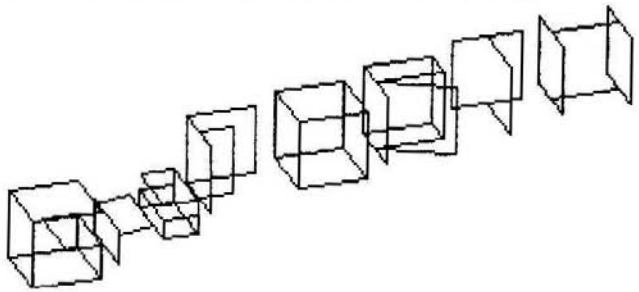


Figura 27.2 — Apresentação dos caracteres GraFORTH em diagonal.

27.4 Salvando o sistema com a nova abertura

Acabamos de montar a nossa nova apresentação do sistema e estamos em condições de guardá-la em disco juntamente com o sistema GraFORTH. Sempre que a carregarmos do disco ela se inicia automaticamente.

Servimo-nos deste exemplo justamente para mostrar a possibilidade (já mencionada anteriormente) de podermos ter um sistema GraFORTH personalizado, com as palavras mais úteis para nós já no instante em que é carregado, podendo inclusive ser definido um sistema para uma aplicação específica.

Inicialmente deve-se colocar no dicionário somente as palavras que você achar úteis e, por último, a abertura anteriormente definida. É aconselhável deixar, pelo menos, as palavras-padrões do sistema (aquelas que estão presentes na versão que você adquiriu), uma vez que será regravado o novo sistema sobre o antigo, perdendo-se as palavras que não estejam nesse último.

Para gravar o sistema, faremos uso da palavra SAVEPRG, que cria um arquivo binário com o sistema:

```
Ready  SAVEPRG
Save File Name :
```


Nossa resposta é OBJ.FORTH, já que queremos que esta versão seja carregada sempre que iniciarmos o sistema.

A seguir devemos responder a pergunta:

```
Autorun (Y/N) :
```

Entrando com a tecla ⟨Y⟩, uma vez que o topo do dicionário (a nossa apresentação que acabamos de criar) deve ser executada automaticamente. Terminado, o novo sistema será gravado no disquete sobre o original.

Agora testemos: entre no EDITOR com EDIT, tecle ⟨ CTRL⟩⟨D ⟩ e ⟨RETURN⟩ e responda ao 'prompt' com PR#6 (ou outro valor, caso a unidade de disco não esteja conectada nesta saída), que corresponde à reinicialização do sistema (equivale a desligar e ligar o micro, porém sem desgastar os componentes eletrônicos). Observe o sistema sendo carregado, executando a abertura e, ao terminar, apresentar o 'prompt' "Ready" na última linha da tela. Se você não desejar acompanhar toda a apresentação, basta acionar qualquer tecla e o sistema passará o acesso diretamente à linguagem, interrompendo a apresentação.

Apêndice

Mapa da memória do Apple II com o sistema GraFORTH

Neste item, listamos os principais endereços úteis da memória do seu Apple II com o sistema GraFORTH habilitado. No quadro apresentado nesse apêndice, temos uma descrição do uso da memória do seu Apple II pelo sistema GraFORTH, independentemente da memória disponível no seu microcomputador.

MAPA DO SISTEMA GRAFORTH

0 a 255 ($0000 a $00FF)	Página zero do 6502:
0 a 31 ($0 a $1F)	Não usado
32 a 79 ($20 a $4F)	Uso do monitor do Apple II
80 ($50)	Ponteiro-1 de texto do GraFORTH (2 bytes)
82 ($52)	Ponteiro-2 de texto do GraFORTH (2 bytes)
84 ($54)	Ponteiro-1 de gráficos do GraFORTH (2 bytes)
86 ($56)	Ponteiro-2 de gráficos do GraFORTH (2 bytes)
96 a 127 ($60 a $7F)	Não usado
128 ($80)	Posição X da última plotagem (1 byte)
130 ($82)	Posição Y da última plotagem (1 byte)
156 ($9C)	Ponteiro da pilha de dados
157 ($9D)	Ponteiro da pilha de retorno
218 a 255 ($DA a $FF)	Área da matriz de trabalho
256 a 511 ($0100 a $01FF)	Pilha do 6502
512 a 767 ($0200 a $2FFF)	Buffer de entrada de linha do sistema GraFORTH
768 a 811 ($0300 a $032B)	Área de trabalho de operações com matriz 3D
812 a 935 ($032C a $03A7)	Pilha de compilação e área da variável string PAD
936 a 975 ($03A8 a $03CF)	Buffer das cores dos gráficos
976 a 1023 ($03D0 a $03FF)	Área de ligação com o DOS
1024 a 2047 ($0400 a $07FF)	Página 1 de texto e de gráficos de baixa resolução do Apple II

2048 a 2815 ($0800 a $09FF) Área de armazenamento da fonte primitiva (principal) de caracteres

2816 a 5887 ($0B00 a $16FF) Área livre para usuários

5888 a 6655 ($1700 a $19FF) Área de dados (de posição e rotação) dos objetos gráficos:
- 5888 a 6143 ($1700 a $17FF) Apagados
- 6144 a 6399 ($1800 a $18FF) Não exibidos (objetos da tela inativa)
- 6400 a 6655 ($1900 a $19FF) Exibidos (objetos da tela ativa)

6656 a 7679 ($1A00 a $1DFF) Tabela de vetores gráficos do GraFORTH

7680 a 7935 ($1E00 a $1EFF) Pilha de dados

7936 a 8191 ($1F00 a $1FFF) Pilha de retorno

8192 a 16383 ($2000 a $3FFF) Página 1 de gráficos de alta resolução do Apple II

16384 a 24575 ($4000 a $5FFF) Página 2 de gráficos de alta resolução do Apple II

24576 a ($6000 a $6EAF) Sistema monitor do GraFORTH

($6EB0 a $82B1) Dicionário de palavras do sistema GraFORTH

($82B2 a $87FF) Área para acréscimo de novas definições de palavras do GraFORTH

-30720 a -28673 ($8800 a $8FFF)
Área de trabalho do editor de textos (ou área para definição de novas palavras se o editor de textos não for utilizado). Com placa de expansão de memória, esta área se estende ao endereço -18945 ($B5FF).

-28972 a -26113 ($9000 a $99FF)
Área usada pelo editor de textos do GraFORTH sem a placa de expansão de memória. Se houver placa de expansão de memória, o editor de textos do sistema GraFORTH será carregado na área -18944 a -16385 ($B600 a $BFFF).

-26114 a -16385 ($9A00 a $BFFF)
A área reservada para o sistema operacional do GraFORTH, caso seu Apple II não possua placa de expansão de memória.

-16384 a -12289 ($C000 a $CFFF)
Área de I/O do Apple

-12288 a -1 ($D000 a $FFFF)
Área da ROM do Apple II (sistema monitor do Apple e Applesoft/ Integer Basic). Se houver placa de expansão de memória, a ROM estará inabilitada, enquanto que um conjunto de 12K Bytes de RAM da placa de expansão estará ativa com o sistema monitor do Apple e o sistema operacional do GraFORTH.

Índice remissivo

Este livro foi impresso
(com filmes fornecidos pela Editora)
na Gráfica Editora Bisordi Ltda.,
à Rua Santa Clara, 54 (Brás),
São Paulo.