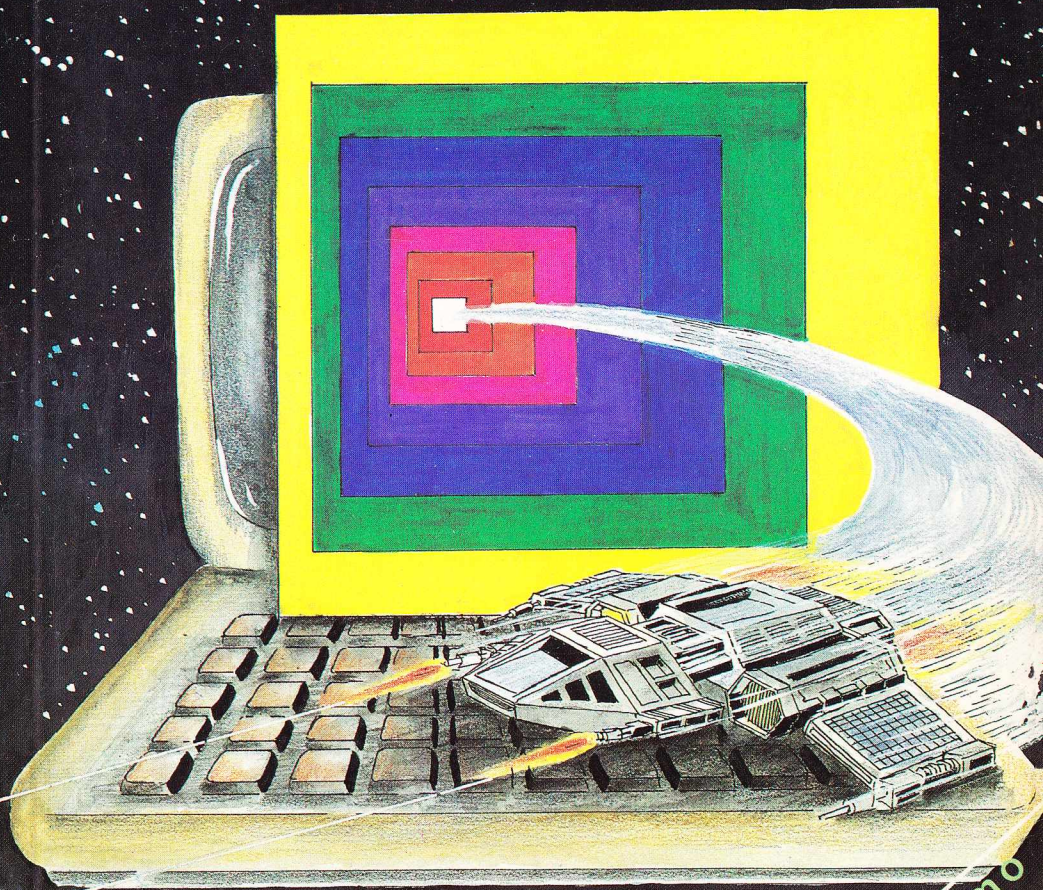


# BRINCANDO COM O TAS COLOR



nobel

VICTOR  
MIRSHAWKA

Compatível com o  
CP 400 e  
COLOR 64



**BRINLANDO  
COM O  
TAS COLOR**

CIP-Brasil. Catalogação-na-Publicação  
Câmara Brasileira do Livro, SP

Mirshawka, Vítor, 1941-  
M655j Brincando com o TRS-Color / Victor Mirshawka.—  
São Paulo : Nobel, 1985.

ISBN 85-213-0292-4

1. Desenho eletrônico 2. Jogos eletrônicos  
I. Título.

85-0343

17. CDD-651.8

18. -001.6425

Índices para catálogo sistemático:

1. Desenho eletrônico : Computadores : Programas : Processamento de dados 651.8 (17.) 001.6425 (18.)
2. Jogos : Computadores : Programas : Processamento de dados 651.8 (17.) 001.6425 (18.)
3. Programação : TRS-Color : Computadores : Processamento de dados 651.8 (17.) 001.6425 (18.)
4. TRS-Color : Computadores : Programação : Processamento de dados 651.8 (17.) 001.6425 (18.)

CAPA E ILUSTRAÇÕES: Lirio Fissao Yuasa

# VICTOR MIRSHAWKA

Professor titular de Cálculo Numérico, Estatística e Pesquisa Operacional da Faculdade de Engenharia da Fundação Armando Álvares Penteado.

Professor titular de Cálculo Numérico e Ciência da Computação, Estatística e Pesquisa Operacional da Universidade Mackenzie.

Professor associado do departamento fundamental da Escola de Engenharia Mauá.

Mestre em Estatística pela Universidade de São Paulo.

# BRINCANDO COM O TAS COLOR

nobel

© 1985 Livraria Nobel S.A.

Livraria Nobel S.A.  
Rua da Balsa, 559  
Cep 02910  
São Paulo - SP

É proibida a reprodução total ou parcial  
desta obra (lei 5998/73),  
sem a permissão por escrito dos editores.

Impresso no Brasil / Printed in Brazil

# Agradecimentos

Foi contínua e indispensável a ajuda da minha querida esposa e amiga Nilza Maria e dos meus dois filhos maiores, Victor Junior e Sergio, na elaboração e correção deste livro.

Cumpre destacar que o terceiro filho Alexandre (Sacha para os íntimos) também aos poucos vai se alfabetizando e manipulando cada vez melhor o microcomputador trazendo com isto dúvidas ou criações que convém explicar ou incluir em um livro...

Sem dúvida nenhuma, sem o auxílio deles não poderia jamais, apresentar em prazo tão curto, uma série de textos distintos quanto ao conteúdo específico, porém todos eles voltados para o uso do microcomputador.

Acredito muito hoje (e o TRS-2| tem sido um bom meio para isto) na frase: "Uma família unida trabalha e se diverte unida..."

Sem isto seria bem difícil escrever este livro.

"Thank you", Toshiro Iqueda, datilógrafo sem igual e amigo de muitos anos, que decifrou os originais e os transformou em mais um livro.

"Arigato" Lirio Fissao Yuasa pela excelente parte gráfica.

"Merci", Angélica Ayres e novamente Toshiro Iqueda, pelo primoroso trabalho de editoração.

Finalmente, sou grato mais uma vez aos amigos da Livraria Nobel S.A., Ary K. Benclowicz e Luigi Zamboni, por acreditarem na validade de mais este trabalho colocando à nossa disposição todo o setor de publicação e divulgação.



Observação Importante O.I.

Vamos chamar no texto de TRS-2 o microcomputador da Tandy Radio Shack (TRS) ou seja o seu Color Computer 2.

Entre os similares nacionais destacamos o CP-400 e o Color 64.





# Apresentação

Se você não sabe nada sobre microcomputadores e gostaria de evitar longas e técnicas explicações sobre o assunto relaxe, pois este é o livro que você esperava!!!

Utilizando este livro como se fosse um manual você estará habilitado a interagir e desfrutar de todos os recursos gráficos do seu TRS-2.

Como o próprio nome diz "BRINCANDO COM O TRS Color" é um livro que contém programas que irão distrair e ao mesmo tempo torná-lo(la) um(a) grande desenhista.

A finalidade básica deste livro é ajudá-lo(la) a descobrir que a habilidade de criar interessantes gráficos é um ponto de auto valorização e você ao terminar de ler o mesmo tenho certeza, terá grande admiração pelo TRS-2 porém por si próprio(a) muito maior ainda!!!

Você poderá na versão mais simples utilizar uma tela de 64 x 32 ou seja 2048 pontos e obter um lindo conjunto de cores e gráficos com pontos relativamente grandes.

Mas, poderá também usar toda a capacidade do C B E (COLOR BASIC estendido) e ter gráficos coloridos em um quadriculado de 256 x 192 ou seja 49152 pontos.

Neste caso você terá os chamados gráficos de alta resolução.

Existe também a possibilidade de se ter gráficos de baixa resolução (quadriculado de 128 x 96 pontos) e de média resolução (quadriculado de 128 x 192 pontos).

As instruções para gráficos coloridos são apresentadas no texto de uma forma gradual de maneira que você possa pausadamente ir ganhando o pleno domínio do que se exhibe na tela a medida que aprende uma nova instrução gráfica ou não gráfica necessária para a elaboração de um programa.

Comenta-se em cada programa o porque de cada instrução e explica-se de forma sucinta a sua função no mesmo.

Você caro(a) leitor(leitora) adquirirá o completo entendimento da capacidade gráfica que se atinge no TRS-2 via CBE ao chegar ao último capítulo, isto tenho confiança plena, desde que use constantemente o livro enquanto estiver em frente do microcomputador.

Experimente fazer as "pequenas tarefas" deixadas para você como exercícios em quase todos os capítulos.

Nestas ocasiões tenha no TRS-2 o seu grande auxiliar.

Caso você cometa erros ele lhe dará indicações sobre isto bastando para tanto recorrer ao Apêndice 1.

Quando esquecer o que significa alguma instrução ou função recorra ao Apêndice 2.

Explore ao máximo o seu TRS-2 !!!

Descubra o que ele pode e não pode fazer.

Gaste todas as suas horas disponíveis, teclando, tocando, apertando, fazendo até que o TRS-2 faça coisas estranhas porém não tenha medo do mesmo.

O TRS-2 se tornará ao final deste livro seu submissor, porém rápido e infalível "servo" e aí sim você terá conhecimento e o que é principal certeza da relação interminável de coisas que ele pode fazer por ti!!!!

# Sumário

Capítulo 1 - CONHECENDO O SET E O RESET .....	1
Capítulo 2 - DESENHOS SIMÉTRICOS OBTIDOS COM O SET ...	15
Capítulo 3 - ROSÁCEAS .....	23
Capítulo 4 - ESPIRAIS .....	29
Capítulo 5 - MAIS RÁPIDO QUE O VENTO .....	33
Capítulo 6 - PERISCÓPIO AMARELO .....	47
Capítulo 7 - BLACKJACK E DEPOIS SUICÍDIO .....	53
Capítulo 8 - CRAPS COM MÚSICA .....	63
Capítulo 9 - O COSMO COLORIDO .....	67
Capítulo 10 - POT-POURRI CIRCUNFERENCIAL .....	77
Capítulo 11 - DESENHANDO ARCOS .....	85
Capítulo 12 - TRAÇANDO LINHAS RETAS COLORIDAS .....	89
Capítulo 13 - TOMANDO A SUA LINHA .....	93
Capítulo 14 - MOLDURAS VAZIAS E COLORIDAS .....	99
Capítulo 15 - ACERTANDO NO PONTO .....	105
Capítulo 16 - PINTANDO O "SETE" .....	111
Capítulo 17 - ACERTANDO NA FIGURA .....	115
Capítulo 18 - A PODEROSA INSTRUÇÃO DRAW .....	121
Capítulo 19 - EXPLORANDO OUTROS CAMINHOS .....	129
Capítulo 20 - SUBCADEIA DENTRO DE UMA LINHA DRAW .....	135
Capítulo 21 - MOVIMENTO RELATIVO .....	139
Capítulo 22 - MEMORIZANDO .....	147
Capítulo 23 - COPIANDO CONFORME O FIGURINO .....	153
Capítulo 24 - PÁGINAS USADAS NA TOTALIDADE .....	167
Capítulo 25 - CANHÃO ROTATIVO .....	171
APÊNDICE 1 - MENSAGENS DE ERRO NO CBE (COLOR BASIC ESTENDIDO) .....	177
APÊNDICE 2 - SUMÁRIO DO C B E (COLOR BASIC ESTENDIDO)	181



# CONHECENDO O SET E O RESET

Um dos mais poderosos recursos do seu TRS-2 é o que permite a obtenção de gráficos coloridos.

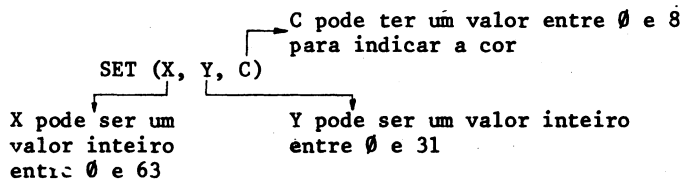
Aliás, este é o assunto primordial deste livro, daí o seu nome "BRINCANDO COM O TRS COLOR".

Nós não desprezaremos as outras qualidades tais como rapidez e precisão nos cálculos, boa apresentação na tela, som para alertar ou para distrair, etc. só que as mesmas serão exaustivamente abordadas nos outros livros sobre o TRS-2 (veja a lista completa dos livros de computação já publicados e a publicar na 4ª capa).

Mas se aqui o assunto é gráficos direi inicialmente que existem duas formas "rudimentares" para elaborar os mesmos.

Uma primeira é utilizando o SET para ativar um ponto na tela e o RESET para "apagar". A segunda forma será apresentada no Capítulo 5.

A forma geral da instrução SET é



As cores do seu TRS-2 estão na Tabela 1.

Código (C)	Cor
∅	preta
1	verde
2	amarela
3	azul
4	vermelha
5	bege
6	ciano
7	lilás
8	laranja

Tabela 1



Observação Importante - O.I.

Cuidado para não usar COLOR no lugar de C pois COLOR é uma palavra reservada no *COLOR BASIC* *estendido*, que nós simplesmente indicaremos por CBE.

As cores podem variar as suas tonalidades de acordo com o seu aparelho de T.V.

O código ∅ (zero) corresponde, na realidade, a ausência de cor.

Na Figura 1 está indicada a disposição da tela se utilizarmos a instrução SET.

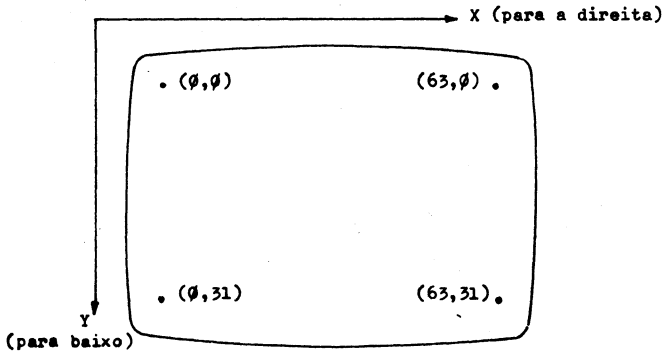


Figura 1

Escrevendo por exemplo SET (28,15,2) estamos ordenando ao TRS-2 que coloque na posição horizontal 28 e na posição vertical 15 um ponto de cor amarela de um retângulo preto maior sobre uma tela verde (fundo verde).

Você deve ter o seguinte aspecto depois de teclar a instrução e pressionar a tecla **ENTER** :

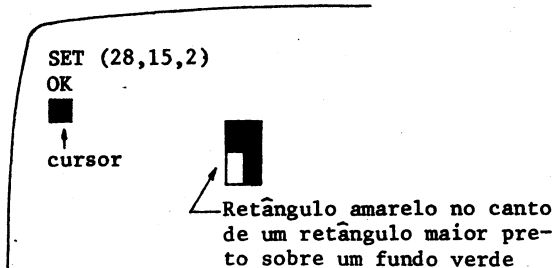


Figura 2

Tecla **CLEAR** para limpar o que está na tela.

Se você agora escrever, ainda a nível de comando, isto é sem número de linha

SET(29,14,2) : SET(28,15,2) e apertar a tecla

**ENTER** terá o aspecto apresentado na Figura 3

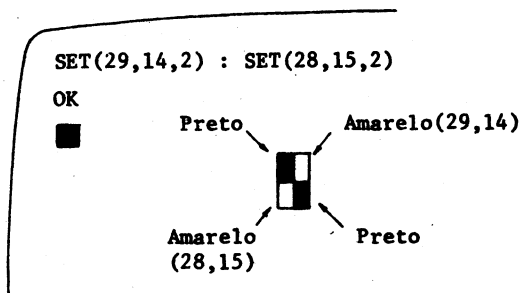


Figura 3

Não aperte **CLEAR** e no lugar disto tecla mais duas instruções SET.

Você terá agora:

SET(29,14,2) : SET(28,15,2)  (já feito antes)  
 SET(29,15,2) : SET(28,14,2)  (novas instruções)

e o que vai ver na tela é o que está mostrado na Figura 4.

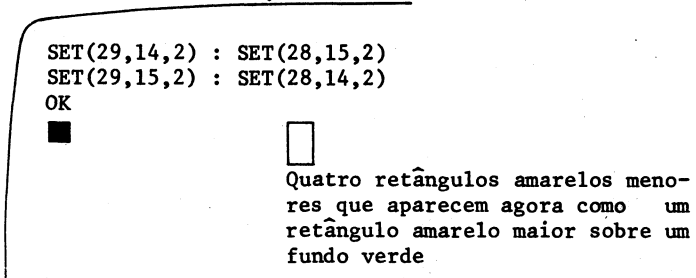


Figura 4

O que realmente ocorreu é que as 4 posições SET [(28,14), (28,15), (29,14) e (29,15)] correspondem a uma única posição PRINT (Figura 5).

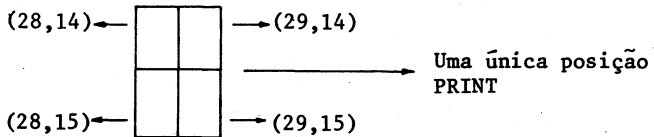


Figura 5

O.I.

- 1) O significado dos dois pontos (:)

Podemos mandar mais que uma instrução para o TRS-2 se as mesmas estiverem separadas por dois pontos (:).

- 2) Você já estava desconfiado(a) que PRINT quer dizer "imprimir" em inglês.

E é isso mesmo!!!

Caso você queira imprimir algo deve usar a instrução PRINT após a qual pode vir o nome de uma variável ou uma mensagem entre aspas.





Um PRINT após o qual não há nada, fará com que saltemos uma linha.

Existe uma outra opção para imprimir e ela ocorre quando se usa a instrução

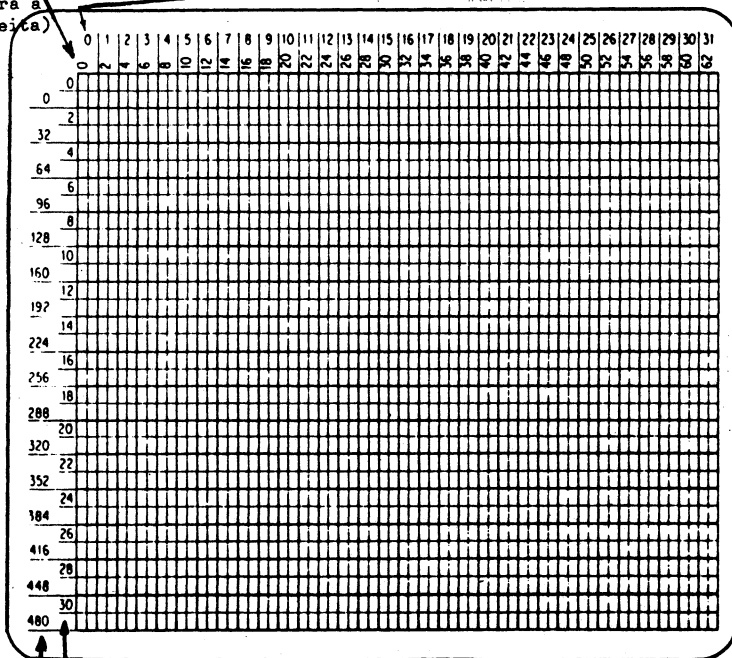
PRINT @ P, Mensagem ou valor de alguma variável

onde P é a posição da tela podendo ser qualquer número entre 0 e 1023. A posição P pode ser indicada por uma expressão numérica.

P varia de 0 a 63 na linha de cima, 64 a 127 na 2ª linha e assim por diante (veja a Figura 6).

Posições SET  
(para a direita)

Posições PRINT @



Posições SET  
(para baixo)

Figura 6

Posições PRINT @

3) Lembra assim que cada posição PRINT tem quatro posições SET no mesmo "quadro".

Usando uma cor e mais a cor preta, pode-se ter 16 caracteres gráficos distintos (veja a Figura 7).

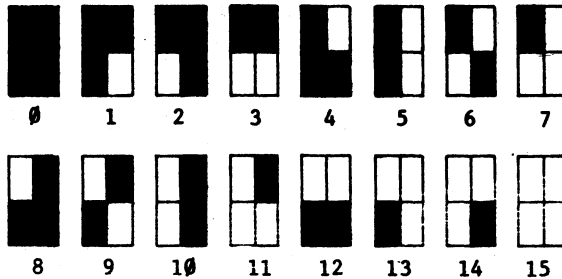


Figura 7

Bem vamos então direto para o 1º programinha que lhe permitirá testar e principalmente ajustar as cores do seu aparelho de T.V., o brilho, a sintonia.

Tecla para começar

```

5 CLS(8)
10 FOR X = 0 TO 63
20 FOR Y = 0 TO 31
30 C = INT (X/8 + 1)
40 SET (X,Y,C)
50 NEXT Y
60 NEXT X
70 GOTO 70
  
```

Execute, isto é tecla um RUN e depois pressione a tecla **ENTER**.

Você deve ver na sua tela 8 faixas verticais nas cores mencionadas na Tabela 1.

Você já notou que ao ligar o seu TRS-2 as letras que bate são pretas com um fundo verde.








Caso isto esteja invertido (verde com fundo preto) aperte e mantenha abaixada **SHIFT** e tecla 0 (zero) para estar no modo normal (fundo verde).

Aliás o TRS-2 possui algumas teclas que não se vê nas máquinas de escrever.

Essas teclas realizam funções especiais, como a

**SHIFT** que acabamos de citar.

A explicação do uso das mais importantes está na Tabela 2.

TECLA	FUNÇÃO
	Retrocesso e limpeza do último caractere teclado
<b>SHIFT</b> 	Modifica o modo de introdução (são maiúsculas/maiusculas e minúsculas)
<b>SHIFT</b> 	Introduz um colchete direito ]
<b>SHIFT</b> 	Introduz um colchete esquerdo [
<b>SHIFT</b> 	Provoca uma pausa no programa, aperte qualquer tecla para continuar
<b>SHIFT</b> 	Introduz uma seta para trás ←
<b>SHIFT</b> 	Limpa a linha que está sendo introduzida e recomeça a introdução
<b>BREAK</b>	Para a operação ou o programa em andamento e prepara o TRS-2 para um novo comando via teclado
<b>CLEAR</b>	Cancela a linha atual, limpa a tela e coloca o cursor (o retângulo ou bloco brilhante) no canto superior da esquerda
<b>ENTER</b>	Permite a introdução de uma linha do programa ou de um conjunto de instruções a nível de comando. O TRS-2 não interpreta nada até que essa tecla seja usada.

Você vai usar muito esta tecla

É a tecla que você mais vai usar!!

Tabela 2

Explicações complementares sobre o primeiro programa

1) O que é programa?

Resp.: É um conjunto de instruções.

Cada instrução tem um número de linha que deve ser um inteiro positivo.

A instrução com o menor número é a que se executa primeiro caso não haja desvios.

Para executar um programa tecla R U N e pressione a tecla ENTER.

2) Qual é o significado de cada linha?

Resp.: Linha 5 - A forma geral desta instrução é CLS(n) (sendo os parênteses opcionais) e com ela manda-se limpar a tela na cor indicada pelo n ( $1 \leq n \leq 8$ ).

No nosso caso particular queremos a limpeza na cor laranja.

Linha 10 e 20 - Aí começam dois ciclos, laços ou LOOPS.

Eles começam sempre pela instrução FOR e terminam com uma NEXT (linhas 50 e 60).

A forma geral do par FOR-NEXT é

```

┌ 20 FOR Z = A TO B STEP C
│   30
│   40
└ 50 NEXT Z
      ↳ passo
  
```

onde Z é a variável contadora, tendo o valor inicial A, o valor limite B e que em cada volta é aumentado ou diminuído de C.

No nosso caso particular o passo (STEP) é um e neste caso ele pode ser omitido, a variável X tomará 64 valores começando em 0 e terminando em 63 de um em um e a variável Y tomará 32 valores de um em um de 0 até 31.

Neste caso temos dois LOOPS um embutido dentro do outro.

O ciclo interno é o da variável Y e o externo é o da variável X. O ciclo interno deve ser executado totalmente para se ter um novo incremento na variável X (ciclo externo).

No caso do nosso programa o corpo do LOOP é constituído pelas instruções de números de linha 30 e 40.

Linha 30 - Temos a função INT

A forma geral da função inteira (INT) é

INT(A)  
 ↳ argumento que pode ser zero,  
 negativo ou positivo

Caso se tenha:

{ PRINT INT(7.38) sairá 7 - maior inteiro não superior a 7.38  
 { PRINT INT(-8.38) sairá -9 - maior inteiro não superior a -8.38

Neste caso já estamos usando o ponto (.) como sendo a nossa clássica vírgula (,) decimal.

No caso da linha 30 os valores de C que serão obtidos variarão de 1 [INT(0/8+1) = 1] a 8 [INT(63/8+1) = 8].

Você não esqueceu ainda que estes eram os códigos para as cores do seu TRS-2, não é?

Linha 40 - Definidos X,Y e C temos condição de ativar pontos da tela na cor ora atribuída a C e na posição (X,Y).

Bem do SET já falamos bastante.

Linhas 50 e 60 - Todo FOR deve terminar com um NEXT como dissemos há pouco.

Linha 70 - A instrução GOTO n é um desvio incondicional para a linha de número n.

No nosso caso estamos desviando para a própria linha e portanto estamos obrigando o programa a ficar para do neste ponto ou seja "congelando" o mesmo na tela.



O.I. - Não vamos dar a teoria completa da linguagem BASIC, pois o intuito é tornar o livro o mais prático possível.

Caso o caro(a) leitor(a) queira maiores detalhes sobre o BASIC como sugestão deve consultar ou o "BASIC sem segredos" ou "Linguagem BASIC" de nossa autoria.

Procuramos entretanto, explicar com detalhes as instruções que aparecem apenas no C B E (Color BASIC estendido)

Além disto, apresentamos no fim do livro o Apêndice 1, no qual aparece de forma sucinta a característica de cada palavra-chave.

Recorra a ele quando esquecer como se comporta algum comando.

3) O que quer dizer (/) na instrução 30?

Resp.: Bem, aí você já tinha desconfiado que é o símbolo para a operação de divisão.

Aliás, para a adição é o (+), para a subtração é o (-) para a multiplicação é o (\*) e a potenciação é o (^).

Tecler o **BREAK** e prepare-se para um novo programa. É conveniente usar agora a instrução **NEW** (novo em inglês).

Com esta instrução apaga-se o programa anterior da memória do TRS-2.

Se você gostou do primeiro programa, pelo menos quanto a beleza das cores tecler as seguintes variantes do mesmo.

*1ª Variante*

```

10 FOR C = 1 TO 8
20 Y = 2*C - 2
30 FOR X = 0 TO 63
40 SET (X,Y,C)
50 SET (X,Y+1,C)
60 NEXT X
70 NEXT C
80 GOTO 80

```

→ Com este programa saem 8 faixas horizontais largas cobrindo metade da tela, uma de cada cor

*2ª Variante*

```

5 CLS 0
10 FOR C = 1 TO 8
20 Y = 4*C - 4
30 FOR X=0 TO 63
40 SET (X,Y,C)
50 NEXT X
60 NEXT C
70 GOTO 70

```

→ Com este programa saem 8 faixas estreitas na tela toda, uma de cada cor

*3ª Variante*

```

10 FOR C = 1 TO 8
20 Y = 4*C - 4
30 FOR X = 0 TO 63
40 FOR I = 0 TO 3
50 SET (X,Y+I,C)
60 NEXT I
70 NEXT X
80 NEXT C
90 GOTO 90

```

→ Com este programa saem 8 faixas largas enchendo toda a tela com as diversas cores

Para você realmente acreditar no efeito da instrução CLS tecle agora o seguinte programa.

```

10 PRINT "PARA SENTIR, QUE EU TRS-2 MUDO A COR DO FUNDO"
20 PRINT : PRINT
30 PRINT "TECLE UM NÚMERO ENTRE 0 e 8"
40 INPUT N
50 CLS(N)
60 FOR K = 1 TO 499
70 NEXT K
80 GOTO 10

```

Antes de mais nada veja que na linha 40 apareceu a nova instrução INPUT.

Toda vez que quiser entrar com informações durante a execução de um programa use a instrução INPUT.

Na tela deve aparecer um sinal de interrogação e o cursor piscando toda vez que aparecer um INPUT.

No nosso caso particular assim que você entrar com um número de 0 a 8 verá a tela ser limpa com uma dada cor pois a instrução INPUT da linha 40 foi atendida e o programa vai para a linha 50.

Qual é o efeito das linhas 60 e 70?

Resp.: Temos aí um laço FOR-NEXT sem corpo ou seja sem nenhuma instrução intermediária.

A função deste ciclo é para dar uma pausa ou seja gerar um atraso para você admirar a tela na cor N e aí voltar novamente a 10 e chegar a linha 40 pedindo uma nova entrada.

Vejamos agora como se pode ter um ponto indo e voltando como se fosse uma rebatida com coeficiente de elasticidade 1.

{	10 CLS(0)	→	você já sabe que com esta instrução teremos uma tela preta.
	20 FOR V=0 TO 31		
	30 SET (31,V,3)		→ que cor é esta?
	40 RESET (31,V)		
	50 NEXT V		
	60 GOTO 10		



Execute e veja que você obtém um ponto que vai caindo e tudo isto graças a instrução RESET da linha 40.

A instrução RESET diz ao TRS-2 para que apague o ponto na posição (31,V) ou no caso geral na posição (H,V).

Como se vê todo ponto ativado na linha 30 é apagado na linha 40.

Ainda mais no RESET não é necessário o 3º parâmetro pois fica-se com a cor do fundo.

Amplie este último programa, fazendo com que o ponto vá descendo e depois vá subindo continuamente.

Adicione as seguintes linhas:

```
60 FOR V = 31 TO 0 STEP -1
70 SET (31,V,3)
80 RESET (31,V)
90 NEXT V
100 GOTO 10
```

Baseando-se nestas últimas informações podemos programar uma "chuva" de pontos.

Aí vai o programa:

```
10 CLS 0
20 FOR H = 0 TO 63
25 FOR V = 0 TO 31 STEP 2
30 SET (H,V,3)
40 RESET (H,V)
50 NEXT V
60 NEXT H
70 GOTO 10
```

Um pequeno trabalhinho para você estimado(a) leitor(leitora).

Será que você consegue elaborar um programa que permita que saia na tela o que está mostrado na Figura 8?

**Sugestão:** Você deverá introduzir a função RND.

A forma geral desta função é  $RND(n)$  e com ela geram-se números aleatórios (ou "randômicos" como dizem alguns...) entre 1 e  $n$  ( $n > 1$ ).

Assim por exemplo se escrevermos  $X = RND(64) - 1$  poderemos ter como resultado um valor inteiro para  $X$  entre  $\emptyset$  e 63.

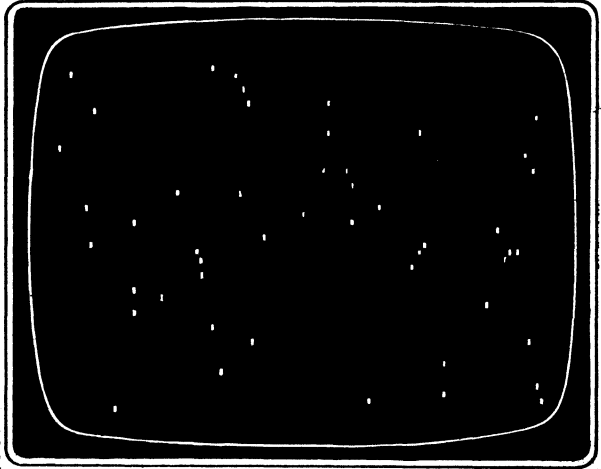


Figura 8

O.I. **AVISOS ÚTEIS**

- 1) NEW apaga a memória.  
Use sempre o comando NEW antes de digitar um programa.
- 2) Para rodar ou executar um programa, tecele RUN e aperte ENTER .
- 3) Para ver todas as linhas de um programa digite LIST e pressione ENTER .  
Caso você queira um trecho apenas do programa tecele por exemplo LIST 10-80 e o TRS-2 lhe mostrará o programa da linha 10 até a 80.
- 4) Para tirar uma linha do programa, tecele apenas o número de linha e pressione ENTER .
- 5) Para mudar substancialmente uma linha do programa tecele-a de novo e ela automaticamente vai no lugar da "antiga".

# DESENHOS SIMÉTRICOS OBTIDOS COM O SET

Inicialmente vamos elaborar um programa que permita obter algo semelhante ao que está sendo mostrado na Figura 9.

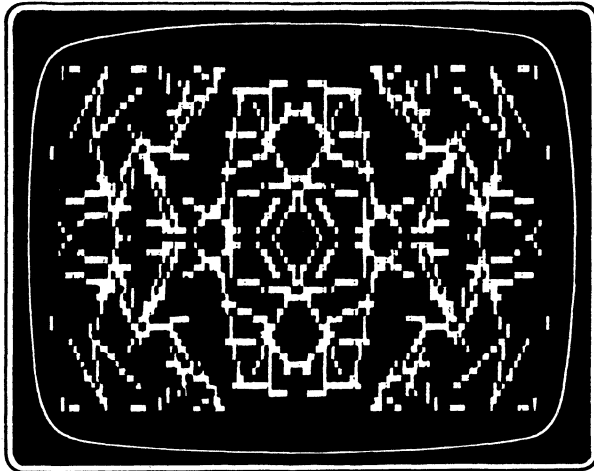


Figura 9

```

10 CLS(0)
20 FOR J=1 TO 30
30 X = RND(64)-1 : Y = RND(32)-1
40 H = 2*RND(3)-1 : K = 2*RND(3)-1
50 FOR I=1 TO 13
60 X = X+H : Y = Y+K
70 IF X > 63 OR Y > 31 THEN 120
80 IF X < 0 OR Y < 0 THEN 120

```

```

90 SET(X,Y,4): SET(63-X,Y,4)
100 SET(X,31-Y,4): SET(63-X,31-Y,4)
110 NEXT I
120 NEXT J
130 W$ = INKEY$: IF W$ = "" THEN 130 ELSE 10

```

### *Explicações*

Linha 30 → Acabamos de falar sobre a função RND e desta forma nesta linha de instruções múltiplas sorteia-se um valor para X entre 0 e 63 e um para Y entre 0 e 31.

Linha 40 → Aí temos os incrementos de X, representado pelo H e de Y representado pelo K.

Linhas 70 e 80 → Agora apareceu a importantíssima instrução IF-THEN.

Uma instrução IF-THEN permite realizar comparações entre números ou STRINGS.

Mas o que é uma STRING?

Resp.: O CBE aceita, como todo BASIC, as variáveis numéricas indicadas por letras ou letras seguidas de números e as variáveis STRING.

Enquanto o conteúdo de uma variável numérica é um número aquele da variável STRING é uma sequência de caracteres que podem ser letras, números, caracteres gráficos, etc.

Os nomes de variáveis STRING terminam obrigatoriamente com o símbolo \$ (cifrão).

Assim na linha 130 temos a variável STRING W\$.

Deve-se usar aspas para definir o que será armazenado em uma variável STRING.

Por exemplo

```

A$ = "NÚMERO 777" → número da sua casa
B$ = "O MARAVILHOSO TRS-2"

```

C\$ = "DESPERTA BRASIL"

D\$ = ""  $\longrightarrow$  está guardado o vazio ou  
seja nada como aliás está  
indicado para W\$ na linha  
130

Como é que a instrução IF-THEN realiza comparações?

Resp.: Em primeiro lugar o seu TRS-2 entende os símbolos de comparação > (maior que), < (menor que), <> (diferente de), > = (maior ou igual a), < = (menor ou igual a).

Desta forma o TRS-2 se habilita a fazer testes relacionais. Com a instrução IF-THEN "pedimos" ao TRS-II para decidir se deve ou não ser feita determinada coisa.

O TRS-2 verifica a informação que se coloca após a palavra chave IF.

Se (IF) a informação estiver correta ou verdadeira o TRS-2 fará o que está se mandando após a palavra chave THEN.

No nosso caso particular o ponto não pode estar fora da tela o que ocorrerá se  $X > 63$  ou (OR)  $Y > 31$  (linha 70) ou ainda se  $X < 0$  ou  $Y < 0$  (linha 80) quando então manda-se desviar o programa para linha 120 com o que volta-se ao ciclo do J e um novo sorteio é feito para X e Y.

Porém neste caso foram feitas duas comparações com o auxílio do operador lógico OR.

As outras operações lógicas aceitas pelo TRS-2 são o AND (multiplicação lógica) e o NOT (inversão lógica).

Na realidade poderíamos fundir as linhas 70 e 80 em uma única ou seja:

70 IF  $X > 63$  OR  $X < 0$  OR  $Y > 31$  OR  $Y < 0$  THEN 120

com o que economizamos um pouco de tempo de datilografia e temos o mesmo efeito anterior.

Na segunda parte da linha 130 aparece o **IF-THEN-ELSE** e a finalidade da palavra ELSE é fazer com que o TRS-2 faça alguma coisa a mais quando a condição não é verdadeira.

E o que ocorre se a informação após o IF for falsa?

Resp.: O TRS-2 passa para a linha que vem imediatamente após a linha do último IF a não ser que se tenha uma instrução IF-THEN-ELSE quando então se a condição for falsa faz-se o que está indicado após o ELSE.

Linhas 90 e 100 → Linhas de instruções múltiplas que ativam pontos simétricos em relação à vertical que passa pelo meio da tela, na cor vermelha.

Linha 130 → Inicialmente temos a palavra INKEY\$.

O INKEY\$ manda o TRS-2 olhar o teclado para ver se você apertou alguma coisa, sem a necessidade de apertar o **ENTER** depois.

O TRS-2 faz isto com tremenda velocidade.

O programa armazena em W\$ o caractere que for pressionado e aí compara com a STRING vazia ou sem nada"".

Como a condição é falsa desvia-se para a linha 10 que é a indicação após o ELSE.

Em outras palavras, enquanto você não apertar nenhuma tecla em W\$ estará armazenado o "nada" e como "nada" é igual a "nada" a condição é verdadeira.

Desta forma como o número após o THEN é 130 o programa ficará parado aí ou seja na linha 130.

No momento em que você tocar em alguma tecla a con-

dição é falsa e aí entra em ação o ELSE (de outro modo ou senão em inglês) fazendo o programa voltar a 10 para uma limpeza da tela e começa a elaborar um novo desenho.

Bem, depois de tanta explicação execute o programa e comprove parte do que foi dito.

Já cansou de olhar para um desenho do tipo que está na Figura 9?

Uuootimo!!!

Tecla então o seguinte programinha:

```

10 CLS0 -----> os parênteses são opcionais como
20 K=1: X=0: Y=RND(31)   você vai comprovar e não precisa
                        escrever CLS(0)
30 IF RND(2)=1 THEN K=-K
40 IF RND(4)=1 THEN X=X+1
50 Y=Y+K
60 SET(X,Y,3): SET(63-X,Y,3)
70 SET(X,31-Y,3): SET(63-X,31-Y,3)
80 IF Y<1 OR Y>30 THEN Y=Y-K
90 IF X<63 THEN 30
100 W$ = INKEY$: IF W$="" THEN 100 ELSE 10

```

Execute e veja se sai algo parecido (após várias tentativas) com o que se mostra na Figura ).

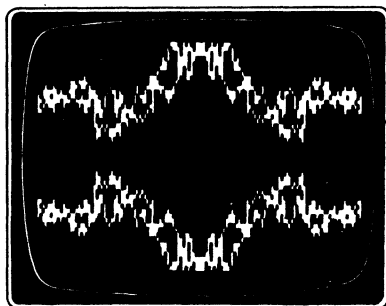


Figura 10

E como última variante do mesmo assunto, não poderia faltar aqui uma versão bem simples para o seu TRS-2 de um programa para o caleidoscópio.

Aí vai o programa:

```

10 CLS0
20 X=RND(32)-1
30 Y=RND(16)-1
40 C=RND(9)-1
50 GOSUB 100
60 GOTO 20
100 IF C=0 OR RND(7) = 3 THEN 160
110 SET(31-X,16+Y,C)
120 SET(31-X,15-Y,C)
130 SET(32+X,16+Y,C)
140 SET(32+X,15-Y,C)
150 RETURN
160 RESET(31-X,16+Y)
170 RESET(31-X,15-Y)
180 RESET(32+X,16+Y)
190 RESET(32+X,15-Y)
200 RETURN

```

Antes de executar o programa vejamos o que significa a instrução da linha 50 ou seja GOSUB 100.

Com esta linha instrue-se o TRS-2 a procurar a subrotina que começa na linha 100.

Mas o que é uma subrotina?

Resp.: É um grupo de instruções de um programa (chamado as vezes de programa principal) com o qual se executa determinada função e ao qual se pode fazer referência várias vezes no mesmo programa, sem a necessidade de escrever novamente a sequência de instruções.



Por que usar subrotinas?

- Resp.: a) Subrotinas diminuem a digitação.  
 b) Subrotinas economizam memória do TRS-2.  
 c) Subrotinas auxiliam a organizar adequadamente um programa.

Como o TRS-2 sabe o lugar correto que deve retornar para o programa principal?

Resp.: Toda a subrotina deve ter como instrução final um RETURN.

É devida a esta instrução que o TRS-2 irá sempre se lembrar que deve sair da subrotina ou seja voltar ao programa principal para a instrução que vem logo após ao GOSUB.

No nosso caso específico atingida a instrução RETURN volta-se para a linha 60 que com um desvio incondicional transfere o programa para a linha 20 onde se sorteia o novo valor de X, aí passa-se para a linha 30 onde se sorteia um novo valor de Y, aí atinge-se a linha 40 onde sorteia-se a nova cor.

Bem, chega-se aí a subrotina (linha 50).

Na linha 100 a primeira instrução é um teste que tanto pode mandar apagar pontos [caso C seja 0 (zero) ou  $RND(7) = 3$ ] ou em caso contrário ativar pontos.

Note que irá ocorrer uma das alternativas e no final de cada uma delas surge uma instrução RETURN (linha 150 e linha 200) fazendo com que se volte a linha 60 e assim por diante.



# ROSACEAS

Chegou a hora de elaborar com o auxílio da instrução SET o gráfico de algumas rosáceas como as indicadas nas Figuras 11 e 12.

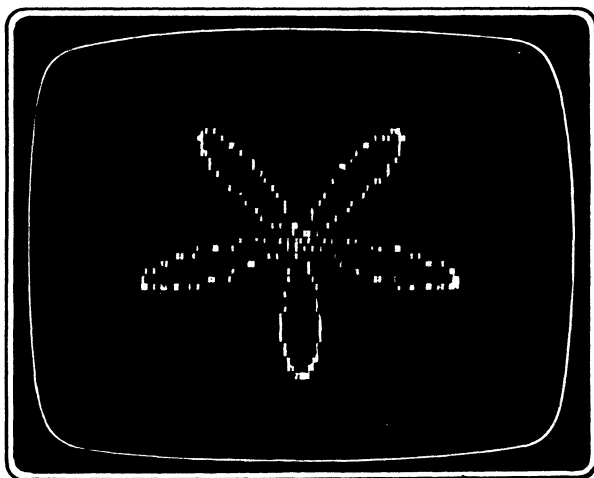


Figura 11 - Rosácea de cinco pétalas

As curvas chamadas rosas de n pétalas são geradas variando-se o raio da figura em questão através das funções trigonométricas seno(SIN) e cosseno(COS).

Especificamente a instrução fundamental é

$$R = M * \text{JIN}(C * A)$$

é o raio →  
 uma constante que torna a figura maior ou menor na tela →  
 função seno →  
 constante que determina o número de pétalas da rosácea →  
 ângulo em radianos →

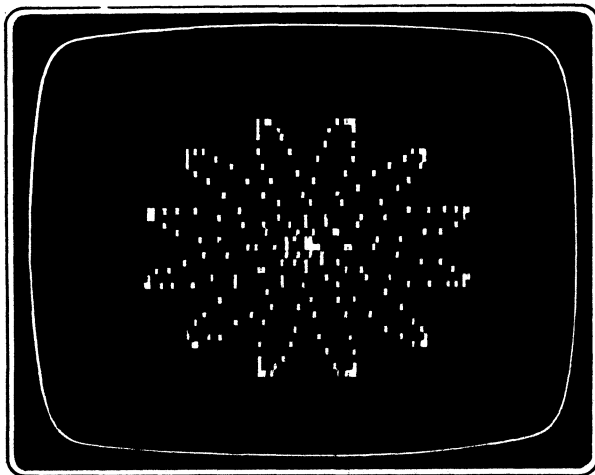


Figura 12 - Rosácea de doze pétalas

Aí vai o programa que permitirá obter as representações das Figuras 11 e 12.

```

5 CLSØ
1Ø C = 5
4Ø FOR A = Ø TO 6.28 STEP .Ø3
5Ø R = 4 * SIN(C*A)
6Ø X = R*7*CGS(A) + 32
7Ø Y = R*3*SIN(A) + 16
8Ø SET(X,Y,4): NEXT A
9Ø GOTO 9Ø
  
```

Execute, isto dê um RUN e veja o que está mostrado na Figura 11.

Mude agora a linha 10 para

```
10 C = 6
```

e veja se sai o que está mostrado na Figura 12.

Caso você queira ter um programa mais geral entre com as seguintes linhas

```
5 CLS 0
10 INPUT "ENTRE COM A CONSTANTE QUE INFLUI NO NÚMERO DE
    PÉTALAS:";C
20 PRINT@ 300,C;: FOR Z = 1 TO 999: NEXT Z
30 CLS 0
90 IF INKEY$="□" THEN 5 ELSE 90
```

└este é o espaço em branco obtido com a barra

Teste o programa agora, entrando com C=9 e veja se sai o que está mostrado na Figura 13.

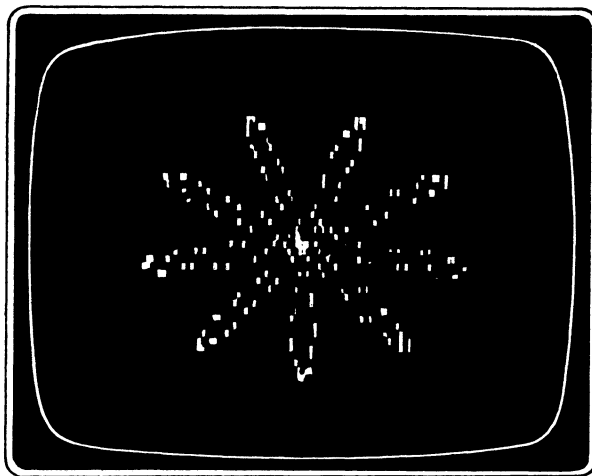


Figura 13

Será que se pode entrar neste programa com valores grandes de C tais como 65, 150, 272?

Resp.: Sim.

Teste com C=60 e veja se sai algo como o indicado na Figura 14.

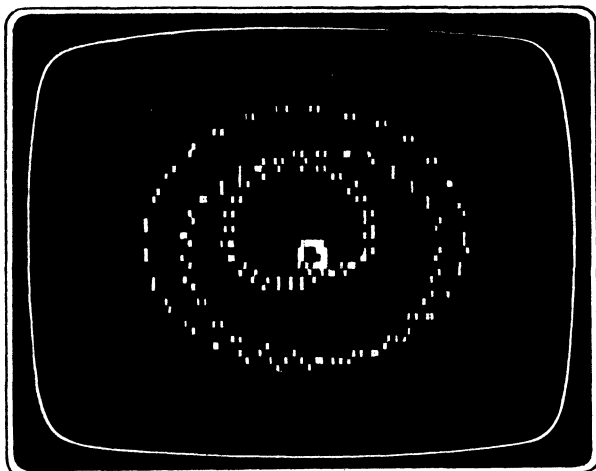


Figura 14

O.I.

1) Qual é a diferença entre INKEY\$ e INPUT?

Resp.: A função INKEY\$ é usada para entrar com informações pelo teclado.

Uai uai, uai mas INPUT não é para isto também?

Correto, mas há uma grande diferença entre as duas ou seja com o INKEY\$ só dá para introduzir um caractere por vez, a tecla ENTER não precisa ser pressionada e o caractere enviado não aparece na tela.

2) As instruções INPUT que começam com mensagem, como é o nosso caso da linha 10, necessitam de um ponto e vírgula (;) antes das variáveis.

Talvez você queira mexer um pouco na equação do raio propriamente dito e neste caso as curvas terão um aspecto totalmente diverso.

Mude a linha 50 para

$$50 \text{ R} = 1 + 3.3 * \text{SIN}(C * A)$$

e entre com C=3 e veja se sai o gráfico da Figura 15.

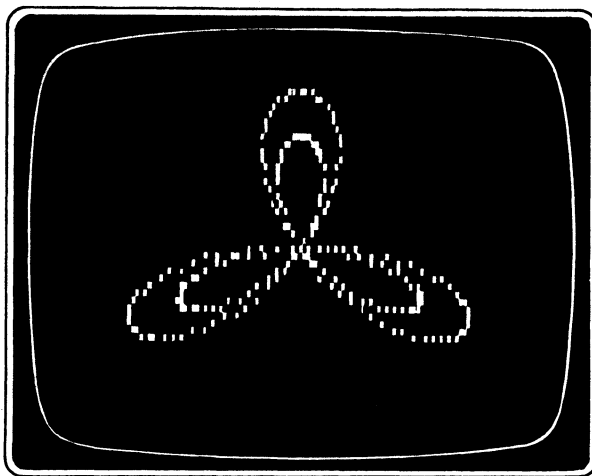


Figura 15

Gostou?

Claro. Entre agora com C=4 e depois com C=5 que vai ver curvas mais estranhas ainda!!!

Tente para finalizar com C=4 e na linha 50 tendo

$$50 \text{ R} = 2 - 2 * \text{SIN}(C * A)$$

Saiu o que está na Figura 16?

Wonderful!

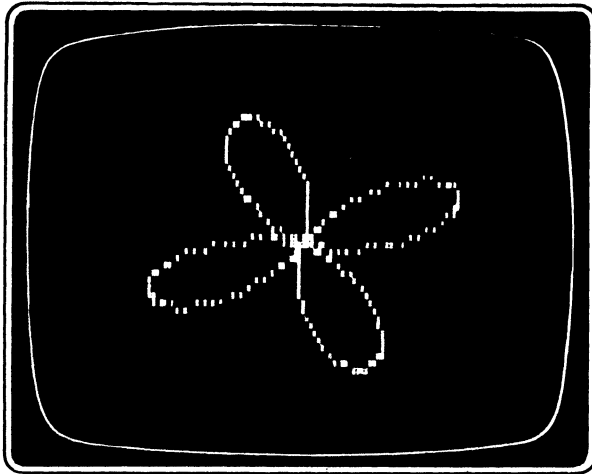


Figura 16

O resto (e é um resto enorme...) é com você feliz proprietário(a) e usuário(a) do TRS-2.

Mude dentro dos limites do possível o C, as funções trigonométricas e deleite-se com o que aparece na tela.

O.I. Não é neste livro que entraremos em grandes elucubrações matemáticas (neste ponto alguns dirão "graças a Deus...") porém a expressão geral no CBE para a função seno é

$$\text{SIN}(A)$$

└ argumento tomado em radianos

e para a função cosseno é

$$\text{COS}(A)$$

└ argumento tomado em radianos

O argumento A pode ser uma expressão numérica.



# ESPIRAIS

Vamos neste momento apresentãr um programa que per<sub>u</sub>mita desenhar uma espiral.

Uma espiral é uma curva aberta que descreve várias voltas em torno de um centro.

Uma espiral é uma curva plana cujo raio polar é uma função constantemente crescente (ou decrescente) do ângulo polar.

Na realidade temos também espirais no espaço porém no TRS-2, por enquanto, o que podemos obter é o que está indicado na Figura 17.

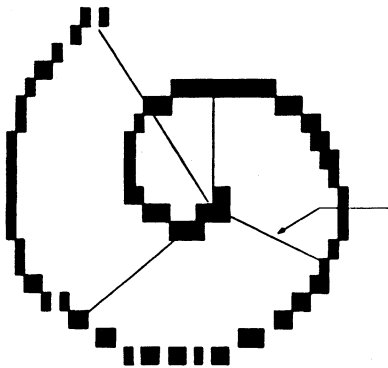


Figura 17

```

10 G = 6
30 CLS
40 PRINT "ESTÁ SE USANDO□"; G; "□ GRAUS": E = G/57.296
45 FOR K=1 TO 500: NEXT K
50 CLS 0 : SET(32,16,4)
60 FOR A=0 TO 40 STEP E
70 X = A*COS(A) + 32: Y=A*SIN(A)*0.42 + 16
80 IF X<0 OR X>63 OR Y<0 OR Y>31 THEN 100
90 SET(X,Y,4)
100 NEXT A
110 GOTO 110

```

Execute e veja se sai o desenho da Figura 18.

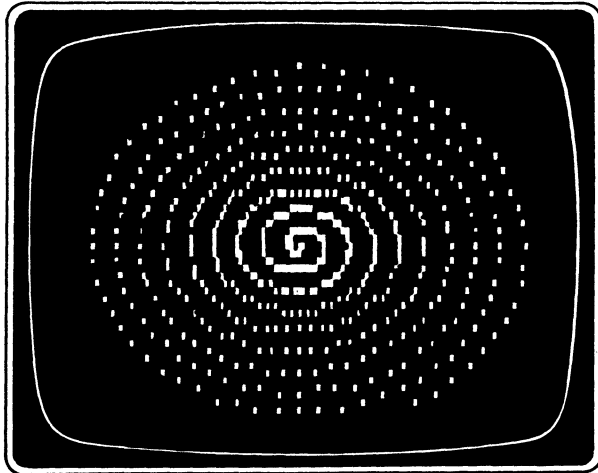


Figura 18

Vamos tornar este programa um pouco mais geral introduzindo as seguintes instruções:

```

10 CLS
30 INPUT "ÂNGULO EM GRAUS:";G
110 FOR K = 1 TO 1700 : NEXT K
120 IF INKEY$="□" THEN 10 ELSE G=G+4: GOTO 40

```

Tente as várias possibilidades e veja se obtém o gráfico da Figura 19.

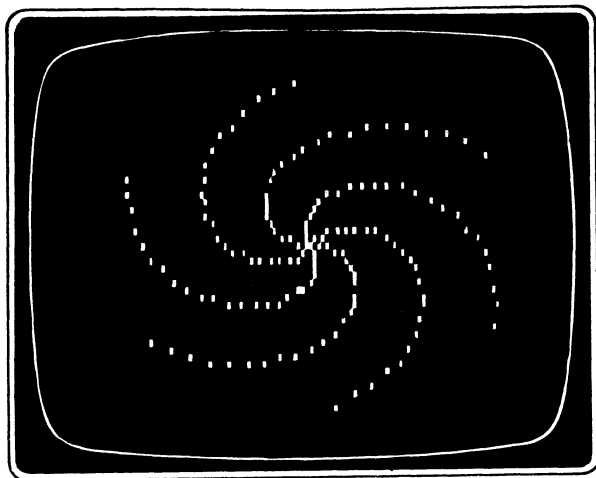


Figura 19

O.I. Efeitos na tela da pontuação no final da linha com a instrução PRINT.



- 1) uma vírgula (,) faz o TRS-2 imprimir em colunas.
- 2) um ponto e vírgula (;) faz o TRS-2 imprimir os números e as cadeias de caracteres (STRINGS) um ao lado do outro.
- 3) sem pontuação o TRS-2 imprime em linhas.

No caso específico da linha 40 o TRS-2 imprime a mensagem "ESTÁ SE USANDO□", a seguir e justaposto por causa do ponto e vírgula, imprime o valor de G e a seguir e justaposto por causa do novo ponto e vírgula a mensagem "□GRAUS".

No texto usaremos "□" para indicar um espaço em branco.



# MAIS RAPIDO QUE O VENTO

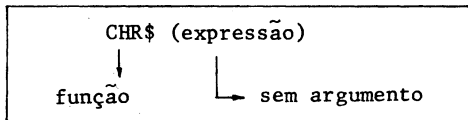
Uma outra forma de se obter gráficos é utilizando a função CHR\$.

Você sabe o que é CHR\$?

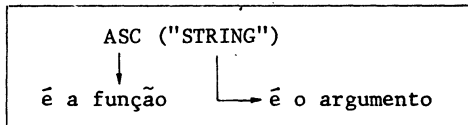
Se não sabe, saiba que com a função CHR\$ converte-se um código para o caractere que ele representa.

Por exemplo, CHR\$(86) converte o código 86 no caractere V (veja a Tabela 3).

O aspecto geral da função CHR\$ é



Uma outra função muito importante é ASC cuja forma geral é



Com esta função obtém-se o código ASCII (American Standard Code for Information Interchange ou seja código padrão americano para intercâmbio de informações).

Um código ASCII é qualquer número inteiro de 0 a 255 na base dez ou o seu equivalente hexadecimal.

Cada caractere do teclado do TRS-2 tem um código correspondente como se mostra na Tabela 3.

Caractere	Código decimal	Código hexadecimal
barra de espaço em branco		
<b>SPACEBAR</b>	32	20
!	33	21
"	34	22
#	35	23
\$	36	24
%	37	25
&	38	26
'	39	27
(	40	28
)	41	29
*	42	2A
+	43	2B
,	44	2C
.	45	2D
/	46	2E
0	47	2F
1	48	30
2	49	31
3	50	32
4	51	33
5	52	34
6	53	35
7	54	36
8	55	37
9	56	38
:	57	39
;	58	3A
<	59	3B
=	60	3C
>	61	3D
?	62	3E
@	63	3F
A	64	40
B	65	41
C	66	42
D	67	43
E	68	44
F	69	45
G	70	46
H	71	47
I	72	48
J	73	49
K	74	4A
L	75	4B
M	76	4C
N	77	4D
O	78	4E
P	79	4F
Q	80	50
R	81	51
S	82	52
T	83	53
U	84	54
V	85	55
W	86	56
X	87	57
Y	88	58
Z	89	59
[	90	5A
]	91	5B
^	92	5C
_	93	5D
BACK	10	0A
CLEAR	8	08
ENTER	9	09
	03	03
	12	0C
	13	0D

Tabela 3

0.I. Os c3digos dos caracteres assinalados com (\*) podem ser obtidos com o aux3lio da tecla SHIFT da seguinte forma

↑ 3 95, ↓ 3 91, ← 3 21, → 3 93 e finalmente  
 CLEAR 3 92.

A Tabela 3 continua pois existem tamb3m c3digos ASCII para as letras min3sculas.

N3s j3 dissemos que estas letras tamb3m podem ser obtidas se apertarmos simultaneamente as teclas SHIFT 0, e a seguir os caracteres desejados.

Caractere	C3digo decimal	C3digo hexadecimal
a	97	61
b	98	62
c	99	63
d	100	64
e	101	65
f	102	66
g	103	67
h	104	68
i	105	69
j	106	6A
k	107	6B
l	108	6C
m	109	6D
n	110	6E
o	111	6F
p	112	70
q	113	71
r	114	72
s	115	73
t	116	74
u	117	75
v	118	76
w	119	77
x	120	78
y	121	79
z	122	7A

Tabela 3 (continua33o)





Aplicando-se na fórmula acima tem-se:

$$CG = 128 + 9 + 16 * 2 = 169$$

Tecele PRINT CHR\$(169) e aperte ENTER

Deve sair:

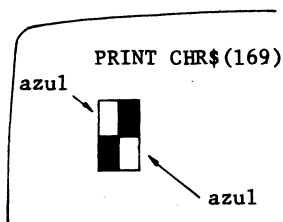


Figura 20

Para começar com as "cosquinhas" nas pontinhas dos seus delicados dedinhos tecele o seguinte mini-programa

```
5 CLS 5
10 PRINT @ 170, CHR$(129);
20 PRINT @ 230, CHR$(130);
30 PRINT @ 290, CHR$(131);
40 PRINT @ 360, CHR$(132);
45 FOR K=1 TO 999: NEXT K
50 GOTO 5
```

Execute e veja se percebe diferentes caracteres gráficos em verde.

Mude os mesmos para cor amarela

"Dica" Basta somar 16 aos números 129,130,131,132.

Vamos agora para um programa mais sutil.

```
10 FOR C=1 TO 7
13 CLS 1
15 PRINT "ESTAMOS NA FAIXA DE"; 128+16*(C-1);"ATE";127+16*C
20 FOR N=0 TO 10
25 CG = N+128+16*(C-1)
30 PRINT @ 159+3*N,N;
40 PRINT @ 224+3*N,CHR$(CG);
```

```

50 NEXT N
60 FOR N=11 TO 15
70 CG = N+128+16*(C-1)
80 PRINT @ 254+3*N,N;
90 PRINT @ 320+3*N,CHR$(CG);
100 NEXT N
110 R$ = INKEY$: IF R$="" THEN 110
120 NEXT C
130 GOTO 130

```

Você sabe por que nas linhas 30,40,80 e 90 termina-se com o ponto e vírgula (;)?

Não, então aí vai a resposta.

O ponto e vírgula faz com que o seu TRS-2 pare de imprimir assim que colocar todos os caracteres gráficos.

Os caracteres gráficos que você viu há pouco nas lindas cores do seu TRS-2 podem ser combinados e armazenados em variáveis STRING.

É este o recurso que vamos utilizar para imprimir o desenho da figura 21.

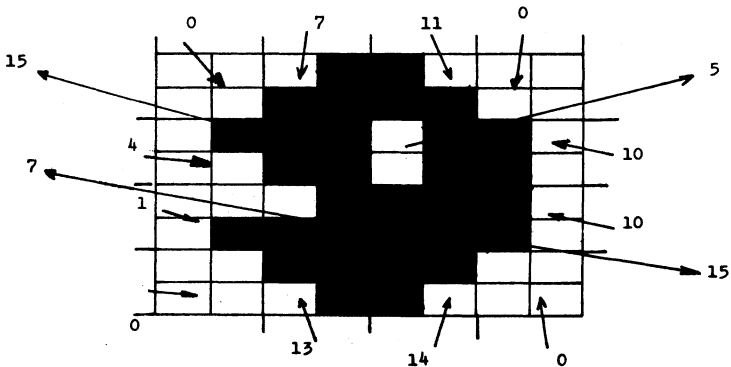


Figura 21 - Neste desenho você deve considerar os caracteres como invertidos em relação a sua apresentação na Figura 7 pois a parte escura é a que deve ter cor e a branca ficará preta.

Aí vai o programa

```

10 FOR C=1 TO 8
20 DC = 128+16*(C-1)
25 CLS0
30 FOR J=1 TO 4
40 READ D1, D2, D3, D4
50 A$(J) = CHR$(DC+D1)+CHR$(DC+D2)+CHR$(DC+D3)+CHR$(DC+D4)
60 PRINT @ 170 + 32*J, A$(J);
70 NEXT J
75 RESTORE
80 R$ = INKEY$: IF R$="" THEN 80
90 NEXT C
100 DATA 0, 7, 11, 0
110 DATA 4, 15, 5, 10
120 DATA 1, 7, 15, 10
130 DATA 0, 13, 14, 0

```

Execute e veja o desenho da Figura 21 em todas as cores.

Explicações sobre as novas instruções e operações que surgiram

Linha 40 → A instrução READ diz ao TRS-2 para encontrar valores para as variáveis D1, D2, D3 e D4 na instrução DATA. Como temos um ciclo que será repetido quatro vezes em cada uma das voltas o TRS-2 "lê" valores diferentes para D1, D2, D3 e D4 e para tornar melhor a leitura do programa colocamos quatro instruções DATA, cada uma contendo quatro números. As instruções READ e DATA podem ser usadas também para ler variáveis STRING (veja no capítulo 7 o programa para o jogo do "21" ou "blackjack").

Qual é a diferença entre INPUT e o READ?

Resp.: A instrução INPUT permite que você introduza diferentes informações cada vez que o programa "roda". Isto é fundamental em certas situações, por exemplo em um jogo, mas deixa as coisas muito lentas...

O seu TRS-2 para, mostra um ponto de interrogação (?), o cursor piscando e fica esperando que você digite a informação.

Com o READ o TRS-2 pega a informação da instrução DATA.

Cada vez que o programa é executado, a informação é lida diretamente da instrução DATA, sem demora e sem paradas.

Caso se queira mudar os dados, basta redigitar a linha DATA com a nova informação.

No nosso caso específico nas instruções DATA enviamos os números que somados com o valor da variável DC permitem obter o  $CG = DC + D1$ ,  $CG = DC + D2$ , etc. ou seja o código do gráfico já na faixa de cor desejada.

Como nós vamos imprimir o desenho da Figura 21 nas 8 cores precisamos restaurar (RESTORE) os dados.

Toda vez que o TRS-2 encontra uma instrução RESTORE (linha 75) fica como se não tivesse lido nada.

O TRS-2 lerá toda a lista outra vez.

O importante das linhas DATA é que você pode colocá-las em qualquer lugar do programa.

Por uma questão de ordenação e talvez de simplicidade de leitura colocaremos ou no fim ou no início de um programa as linhas de instrução DATA.

Linha 50 → AÍ aparece uma variável um tanto quanto estranha, trata-se de A\$(J).

Inicialmente deve-se destacar que existem as variá-

veis simples tais como X, Y2, B\$, G\$, etc. e as variáveis indexadas ou elementos de uma matriz tais como A(3), C(15), A\$(2), N\$(13), etc.

Uma matriz é um conjunto de elementos indexados cada um tendo o mesmo nome.

É por isto que costuma-se também usar o nome de variável coletiva.

Uma matriz tem dimensões podendo ser unidimensional, bidimensional, tridimensional, etc. e além disto cada dimensão tem uma grandeza.

Quando forem usadas matrizes ou seja variáveis coletivas sejam elas numéricas ou STRING é necessário o uso da instrução DIM indicando-se dentro dela a dimensão da matriz.

A instrução DIM deve aparecer no início do programa.

Assim se for escrito

10 DIM A(50), N\$(13)

isto indicará ao TRS-2 que se reserve espaço na memória para 51 elementos de A desde A(0) até A(50) e 14 elementos da variável coletiva STRING N\$ desde N\$(0) até N\$(13).

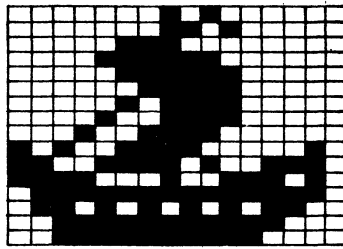
Geralmente não se usa a posição inicial e a instrução DIM pode não ser usada se o índice máximo de uma matriz não for superior a 10.

É o que ocorre no nosso caso particular, pois o maior índice é 4 ou seja temos os elementos A\$(1), A\$(2), A\$(3), A\$(4).

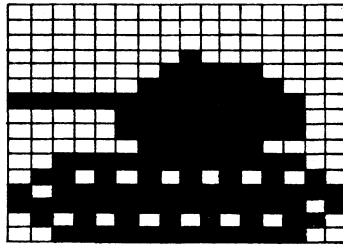
Não esqueça então que o maior número de elementos que uma variável indexada pode ter sem a instrução DIM é 11, de índices de 0 a 10.

Finalmente note na linha 60 que a posição do início de impressão depende do índice da variável indexada.

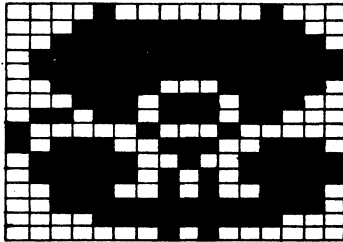
Será que agora você consegue desenhar o que está indicado nas Figuras 22 a), b) e c)?



a)



b)



c)

Figura 22

Se conseguir que na tela do TRS-2 saia por exemplo o tanque da Figura 22b) tente movimentá-lo.

O que, você não aprendeu isto ainda!?!?!?

Então, para melhorar os seus conhecimentos vou lhes apresentar um programa que desenha um trem estilizado usando o SET.

Quem sabe, se com esta "dica" você consegue também fazer o tanque andar.

Vamos fazer um céu ciano (cada um coloca a cor no céu que quiser...) e por isto, aí vai:

```
10 CLS 6
```

A grama deve ser verde e por isto:

```
20 FOR H=0 TO 63
```

```
30 FOR V=22 TO 31
```

```
40 SET(H,V,1)
```

```
50 NEXT V: NEXT H
```

veja se aqui o  
TRS-2 também  
aceita 50 NEXT V,H

Os trilhos são obtidos com:

```
60 FOR H=0 TO 63 STEP 2
```

```
70 RESET (H,22)
```

```
80 NEXT H
```

Finalmente o trem é desenhado com este trecho do programa:

```
90 FOR V=20 TO 21
```

```
100 FOR H=0 TO 13
```

```
110 SET(2+H,V,3): SET(18+H,V,3)
```

```
120 NEXT H,V
```

Com isto, tem-se uma composição ferroviária com apenas dois vagões, cada um feito com 14 pontos de comprimento (0 a 13).

Um começa na posição horizontal 2 e outro na posição 18.

Tem-se espaço em branco entre os vagões.

Execute o programa como está até agora para ver se "bate" tudo...

Bateu (o programa e o resultado)?

Uuoooootimo!!!

Então vamos para o tchi-tchum,tchi-tchum,tchi-tchum  
tã,tã,tã,...

Será que é este o barulho do trem?

vã na Tab.3 e veja que isto é  
↑ apertar a tecla  $\boxed{\rightarrow}$

```

130 R$ = INKEY$: IF R$=CHR$(9) THEN GOSUB 500
140 GOTO 130
500 REM SUBROTINA PARA IR PARA FRENTE
510 L = 0
520 FOR V = 20 TO 21
530 FOR H = 0 TO 1
540 SET(2+L+H,V,6): SET(18+H+L,V,6)
550 SET(16+L+H,V,3): SET(32+H+L,V,3)
560 NEXT H: NEXT V
570 L = L+2
580 IF L > 30 THEN RETURN
590 GOTO 520

```

o que significa este 6?

### *Explicação importante*

Na linha 540 os primeiros pontos de cada vagão são ativados para a cor do céu e com isto parece que são apagados.

Com a linha 550 ativa-se um bloco a frente de cada vagão, sendo que no primeiro movimento são os blocos das posições  $16 = 2+14$  e  $32=18+14$  e isto com a cor do trem que é 3 ou seja azul.

Não esqueça que para que o seu trem ande você deve apertar a tecla  $\boxed{\rightarrow}$ , o que equivale a ter CHR\$(9) (linha 130).





O.I.

Na linha 500 apareceu uma instrução de comentário REM (em inglês "remark" significa comentário).

As instruções REM são colocadas em um programa para que você perplexo(a) leitor(a) tenha mais facilidade de entendê-lo.

Estas instruções não são mostradas na tela durante a execução do programa porém aparecem na listagem.

Se o seu programa já está no TRS-2 tecele o comando LIST e ele lhe mostrará inclusive a linha 500.

Nos programas que vem a seguir usaremos com mais freqüência a instrução REM para explicar dentro do próprio programa o que se quer com cada trecho.



# PERISCOPIO AMARELO

É noite, poucas estrelas no céu e o mar azul está bravio.

Um submarino, devido a problemas no sistema de comunicação está perdido e com o periscópio amarelo levantado está tentando achar seu "rumo".

Com o auxílio da instrução SET e usando também a função CHR\$ vamos fazer um lindo "cocktail" gráfico apresentando algo semelhante ao que se mostra na Figura 23.

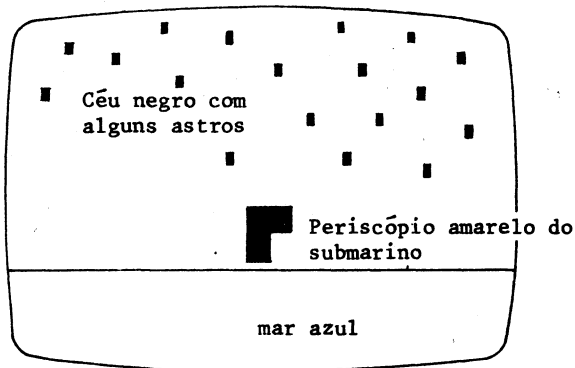


Figura 23

Vamos escrever um programa que permita ao periscópio aparecer numa posição escolhida aleatoriamente e movimentar-se em torno do seu próprio eixo ou seja girando.

Depois o periscópio irá movimentar-se para a direita e depois para a esquerda.

O primeiro trecho do programa permite ativar as estrelas ou outros astros ou quem sabe até um satélite...

```

10 CLS 0
20 FOR S=1 TO 20
30 X=RND(62): Y=RND(16)
40 C=RND(8)
50 SET(X,Y,C)
60 NEXT S

```

Aqui vale qualquer cor

Nós colocamos os "astros" multicoloridos na metade superior da tela de forma que não haja o perigo de uma "colisão" entre o periscópio e algum astro...

Agora vamos pintar o nosso bravo mar azul da posição de impressão 352 até 511.

Isto pode ser feito de 3 maneiras diferentes.

### 1ª Maneira

```

70 FOR MAR=352 TO 510
80 PRINT @ MAR, CHR$(175);
90 NEXT MAR
100 SET(62,30,3): SET(63,30,3)
110 SET(62,31,3): SET(63,31,3)

```

Este é o caractere gráfico em azul



### EXPLICAÇÃO DESTE TRECHO

O laço FOR-NEXT (linhas 70 a 90) coloca o caractere gráfico (código ASCII 175) nas posições de impressão 352 até 510.

Nas linhas 100 e 110 ativa-se a última posição impedindo que haja o "deslizamento" da tela que ocorre se usarmos a instrução PRINT @511.

É por isto que se usou 4 vezes a instrução SET e você não deve ter esquecido que 4 posições SET correspondem a uma posição PRINT e neste caso à última ou seja do canto inferior da direita.

2ª Maneira

```

70 FOR MAR = 352 TO 510
80 PRINT @ MAR, CHR$(175);
90 NEXT MAR
100 POKE 1535, 175

```

## EXPLICAÇÃO DESTE TRECHO

Na linha 100 está a instrução POKE cujo formato geral é

POKE endereço, código ASCII

Com o POKE (que em inglês quer dizer colocar) pode-se por "coisas" na memória do seu TRS-2

As posições da memória de 1024 a 1535 correspondem, no seu TRS-2, as posições de impressão na tela de 0 até 511.

Para obter a posição da memória que corresponde a posição de impressão na tela basta somar 1024 a posição na tela.

Assim tem-se:

localização na memória	=	posição na tela	+	1024
---------------------------	---	--------------------	---	------

Conclusão: O canto inferior da direita é então  $1024 + 511 = 1535$  e quando fazemos POKE 1535, 175 vai para a tela e exatamente no canto inferior da direita, o bloco azul [CHR\$(175)] que está na memória na localização 1535.

3ª Maneira

Como a 2ª Maneira funcionou bem vamos usar a fórmula acima que relaciona a localização na memória com a posição na tela e obter uma forma simples de pintar o nosso mar.

```

70 FOR MAR = 352 TO 511
80 POKE MAR + 1024, 175
90 NEXT MAR

```

Bem, isto já está explicado pelo que foi dito há pouco...

Vamos agora ao trecho que faz aparecer o periscópio em uma posição aleatória e permite girar o mesmo em torno do próprio eixo.


```

120 P = 327 + RND(16)
130 FOR K=1 TO 13
140 PRINT @P, CHR$(158);
150 R=RND(400): GOSUB 1000
160 PRINT @P, CHR$(157);
170 R=RND(400): GOSUB 1000
180 NEXT K

```

Qual é a função de CHR\$(158) e CHR\$(157)?

Resp.: Consulte a Figura 7 e a Tabela 3 e o que vem logo após a Tabela 3.

Agora vem o trecho que move o periscópio (CHR\$(158)  ) para a direita.

```


190 C$ = CHR$(158)
200 AD = RND(350-P)
210 FOR D=1 TO AD
220 PRINT @P, CHR$(128);
230 P = P+1
240 PRINT @P, C$;
250 R=70: GOSUB 1000
260 NEXT D

```

O que se obtém com isto?

Resp.: Veja as explicações abaixo

*Explicações sobre este trecho*

Linha 190 → Armazena-se o caractere gráfico  em C\$.

Linha 200 → Sorteia-se a variável AD (a que distância).


Linhas 210 a 260 → Neste trecho o periscópio amarelo move-se da sua posição atual P para a direita até atingir a posição P+AD.

Em cada passagem do ciclo FOR-NEXT da variável controladora D apaga-se a posição atual do periscópio com o auxílio da linha 220.

Na linha 230 muda-se a posição do periscópio deslocando uma unidade para a direita e na linha 240 desenha-se o mesmo na nova posição.

Bem aí vai o trecho final que permite o movimento para a esquerda e que inclui a subrotina com a qual se dá uma pequena ou razoável pausa (depende do caso).

```

270 C$ = CHR$(157)  → 
280 AD = RND(P-321)
290 FOR E=1 TO AD
300 PRINT @ P, CHR$(128);
310 P = P - 1  →
320 PRINT @ P, C$;
330 R=70: GOSUB 1000
340 NEXT E
350 GOTO 190
1000 FOR Z=1 TO R: NEXT Z
1010 RETURN

```

Devido a esta instrução se vai para a esquerda a partir da posição atual P a uma distância máxima AD

Que tal você modificar um pouco o programa fazendo com que os astros brilhem, desapareçam, surjam constelações, etc.

Se é uma boa idéia, mãos a obra...





# BLACKJACK E DEPOIS SUICIDIO

Aqui vai uma adaptação para o seu TRS-2 do famoso jogo do "21" ("blackjack").

Ele é jogado em cassinos, em roda de amigos numa base amistosa (as vezes não muito amistosa...) e agora contra o seu TRS-2 sem possibilidade de agressão e tão pouco de tornar-se milionário(a).

Você sabe as regras do "21" (não estou me referindo ao popular "racha" de basquete e sim ao jogo de cartas)?

Já esqueceu!?!?!

Mas o que é isto!?!?!

O objetivo do jogo é bem simples

Você precisa obter uma mão de cartas com um total o mais próximo possível de 21 pontos sem ultrapassar este valor.

Caso você ultrapasse este valor (21) você estourou ou seja perdeu o jogo a não ser que...

Se o seu "score" é de 21 pontos ou menos o carteador, que neste caso é o seu TRS-2, mantém uma conversação contigo perguntando-lhe se deseja mais cartas.

Note que o valor das cartas é o seguinte: as figuras todas (valete, dama, rei) valem 10 pontos, o ás 11 pontos e as outras cartas o valor mostrado.

Aliás o TRS-2 mostra na tela as suas duas primeiras cartas viradas (só faltava não mostrar...) e mostra apenas uma das duas dele.

Aproveitando a qualidade de poder exibir cores, os naipes das cartas aparecem nas cores amarela, azul, vermelha e lilás.

O máximo de cartas que você jogador (jogadora) pode ter na mão é cinco e o número máximo de cartas do TRS-2 é duas (uma simplificação nossa para o jogo...).

As cartas viradas apresentam no meio das mesmas o seu valor ou a sua denominação do caso de se ter figura.

Caso você, caro(a) humanóide, tenha uma mão de cartas que inclui um ou mais ases e a soma for superior a 21 pode evitar o "estouro" mudando o valor de cada ás de 11 para 1 ponto.

Esta será uma grande vantagem concedida a você, visto que o ás do TRS-2 valerá sempre 11 pontos.

Não esqueça também que uma figura tem probabilidade 4 vezes maior do que qualquer outra carta de sair.

Daremos mais algumas explicações do programa após o mesmo ter sido executado por você, mas primeiro você deve teclar o seguinte:

```

5 REM VAMOS ESPECIFICAR AS DIMENSÕES
10 DIM G$(5), N$(13), D(52), P(5), C(5)
20 DATA 16, 32, 48, 96, 1
30 DATA **AS**, *DOIS*, *TRÊS*, QUATRO , *CINCO , *SEIS*,
   *SETE*, *OITO*, *NOVE*, *DEZ**, VALETE, *DAMA*, *REI**
40 FOR I=1 TO 5: READ N: G$(I) = CHR$(143+N): NEXT I
50 FOR J=1 TO 13: READ N$: N$(J)=N$: NEXT J
60 CLS 6
70 PT = 0: CT = 0
80 FOR K=1 TO 5: P(K)=0: C(K)=0: NEXT K
90 FOR X=1 TO 52: D(X)=X: NEXT X
100 FOR X=1 TO 5: GOSUB 1000: P(X)=Y: NEXT X
110 FOR X=1 TO 2: GOSUB 1000: C(X)=Y: NEXT X
120 REM AÍ ESTÁ, O QUE TEM O(A) JOGADOR(A) NA MÃO

```

```

130 L=257
140 FOR M=1 TO 2: C=P(M): GOSUB 500: PT = PT+T: NEXT M
150 FOR M=1 TO 3: N=5: GOSUB 1500: NEXT M
160 REM AGORA SE APRESENTAM AS CARTAS DO TRS-2
170 N=5: GOSUB 1500
190 C=C(2): GOSUB 500
200 CT=CT+T
210 PRINT @ 8, "CARTAS DO TRS-2";
220 PRINT 260, "SUAS CARTAS HUMANÓIDE";
230 L=269: K=3
240 PRINT @ 228, "VOCE QUER OUTRA CARTA (S/N)?";
250 R$=INKEY$: IF R$="" THEN 250
260 IF R$ = "N" THEN 340
270 C=P(K): GOSUB 500
280 PT = PT+T
290 FOR J=1 TO K
300 IF PT > 21 AND (P(J)-1)/13 = INT((P(J)-1)/13) THEN
    PT = PT-10: P(J) = 0
310 NEXT J
320 IF PT > 21 THEN PRINT @ 480, "É UMA PENA, MAS VOCE
    ESTOUROU!!";: GOTO 400
330 K=K+1: IF K < 6 THEN 240
340 L=10
350 C=C(1): GOSUB 500: CT = CT+T
360 IF PT <= CT THEN 390
370 PRINT @ 480, "PARABENS HUMANOIDE VENCEDOR(A)";
380 GOTO 400
390 PRINT @ 480, "ESTA VEZ VOCE PERDEU...";
400 IF INKEY$ = "" THEN 400
410 PRINT @ 228, "QUER OUTRA PARTIDA (S/N)? □";
420 R$ = INKEY$: IF R$ = "" THEN 420
430 IF R$ = "S" THEN 60 ELSE END
500 GOSUB 2000: GOSUB 1500: GOSUB 1750: RETURN
1000 Y = RND(52)

```

```

1010 IF D(Y) = 0 THEN 1000
1020 D(Y) = 0
1030 RETURN
1500 L1 = L
1510 FOR J=1 TO 6
1520 L1 = L1 + 32
1530 FOR S = 1 TO 5
1540 PRINT @ L1 + S-1, G$(N);
1550 NEXT S,J
1560 L1 = 0: L = L+6
1570 RETURN
1750 L1 = L-6
1760 FOR J=1 TO 6
1770 L1 = L1 + 32
1780 PRINT@L1+2, MID$(N$(V), J,1);
1790 NEXT J
1800 L1 = 0
1810 RETURN
2000 N = INT((C-1)/13) + 1
2010 V = C - 13*N + 13
2020 IF V=11 OR V=12 OR V=13 THEN T=10 ELSE T=V
2030 IF V=1 THEN T=11
2040 RETURN

```

### Explicações sobre o programa

#### I) Variáveis utilizadas

*Variáveis numéricas simples*

N → define o naipe das cartas

PT → o seu total de pontos

CT → o total de pontos do TRS-2

L, L1 e S → importantes para o PRINT @

T → pontos de uma carta

V → valor da carta (de 1 a 13) em função de uma ordenação

- C → número da carta (de 1 a 52)  
 Y → número entre 1 e 52 escolhido aleatoriamente  
 K → no trecho 230 a 330 é a variável que permite pedir nova carta

*Variáveis numéricas coletivas*

- P ( ) → guarda a carta da pessoa  
 C ( ) → guarda a carta do microcomputador  
 D ( ) → guarda-se a posição das cartas

*Variável STRING simples*

- N\$ → armazena o nome das cartas

*Variável STRING coletiva*

- G\$ ( ) → armazena os naipes representados pelas cores

II) *Instruções novas*

Linha 430 — No final desta linha aparece um END. É evidente que ele manda o TRS-2 terminar o programa.

A instrução END com a finalidade específica de terminar o programa não é necessária, pois o programa para quando atinge a última linha a ser executada.

Em certas situações ou formas de apresentar programas, a instrução END é entretanto bastante útil.

Linha 1780 — Aí aparece a função STRING MID\$

Vamos aproveitar a ocasião e apresentar quatro funções STRING e não apenas a MID\$.

Em primeiro lugar função STRING é aquela cujo argumento é uma STRING.

**Função LEN\$** → Permite obter o número de caracteres de uma STRING

Caso 1º N\$ = "VICTOR MIRSHAWKA"  
 2º PRINT LEN(N\$)  
 16

O comprimento de N\$ é 16 ou seja N\$ tem 16 caracteres incluindo o espaço em branco

**Função LEFT\$** → Permite obter a parte da esquerda de uma STRING

Caso 1º N\$ = "ALEXANDRE MIRSHAWKA"  
 2º PRINT LEFT(N\$,3)  
 ALE

A partir da esquerda os 3 primeiros caracteres são ALE ("left" em inglês quer dizer esquerda)

**Função RIGHT\$** → Permite obter a parte da direita de uma STRING

Caso 1º N\$ = "SERGIO MIRSHAWKA"  
 2º PRINT RIGHT(N\$,5)  
 HAWKA

A partir da direita obtêm-se os cinco últimos caracteres ("right" em inglês quer dizer direita).

**Função MID\$** → Permite obter uma subSTRING de outra STRING

Caso 1º N\$ = "NILZA MARIA MIRSHAWKA"  
 2º PRINT MID\$(N\$, 13,4)  
 MIRS

Seleciona 4 caracteres a partir do décimo terceiro da esquerda para a direita

O.I. Caso você não tenha entendido as definições entre no seu TRS-2 e execute cada um destes programinhas.



Aliás faça uma coisa melhor  
 Bata o seguinte programa

```

10 INPUT "SEU NOME OU UMA PALAVRA COMPRIDA";N$
20 PRINT LEN(N$)
30 PRINT LEFT$(N$,4)
40 PRINT RIGHT$(N$,7)
50 PRINT MID$(N$,9,5)
60 PRINT LEFT$(N$,LEN(N$))
70 PRINT MID$(N$,LEN(N$),1)

```

O seu nome completo ou a palavra deve ter no mínimo 9 caracteres.

Bem com toda essa informação dá para entender que o que se quer na linha 1780 é imprimir em cada volta do laço FOR-NEXT das linhas 1760 a 1790 uma letra do nome da carta que está sendo mostrada.

### III) Trechos destacados do programa

Linha 10 → Indica-se as dimensões das variáveis indexadas.

Como já dissemos anteriormente, poderíamos ter omitido as dimensões para G( ), P( ) e C( ) por serem menores que 11.

Faça isto e confirme que o programa funciona a contento.

Linha 20 → Dados para os naipes representados pelas cores.

Linhas 40 e 50 → Armazenamento dos naipes e dos nomes das cartas.

Linha 70 → Inicialização dos seus pontos e do TRS-2

Linhas 80 - 110 → Distribuição das cartas.

Linhas 120 - 150 → A instrução REM (linha 120) explica isto.

Linhas 160 - 200 → A instrução REM (linha 160) explica isto.

Linhas 210 - 260 → Mensagens e conversação.

Linha 300 → O importante uso do operador lógico AND dentro da instrução IF-THEN permitindo aos ases valer 1 em certas situações.

Linhas 230 - 330 → É aí que está o trecho que permite a você ter mais do que 2 cartas até o limite de 5 cartas.

Linhas 340 - 350 → É mostrada a 2ª carta do TRS-2

Linha 500 → É uma subrotina que tem dentro de si mais três subrotinas.

A subrotina que começa em 500 é chamada várias vezes e representa uma grande economia na digitação.

Linhas 1000-1030 → É uma subrotina que não permite que uma carta apareça mais de uma vez.

Linhas 1500-1570 → É uma subrotina que desenha as cartas sem os nomes.

Linhas 1750-1810 → É uma subrotina que permite imprimir os nomes das cartas.

Linhas 2000-2040 → É uma subrotina que permite obter o valor correto de cada carta e além disto define o seu naipe.

Poderíamos dar mais detalhes, mas acho que estes já são suficientes e quem teclou, executou e jogou o "21" contra o TRS-2 não deve querer mais nada por ora...

Ah! Talvez queira tentar um jogo mais perigoso do tipo roleta russa praticada no TRS-2

Bem aí vai um "programa-sugestão".



```

5 FOR K=1 TO 10
10 PRINT "GIRE O TAMBOR E ENTRE COM UM NÚMERO DE 1 A 10"
20 INPUT X
30 IF X = RND(10) THEN 120
40 SOUND 200,1
50 PRINT "--CLICK--"
60 NEXT K
70 CLS
80 PRINT @ 198, "PARABENS!!!! PARABENS!!!!"
90 PRINT @ 266, "VOCÊ SAFOU-SE DESTA,"
100 PRINT @ 295, "VAI VIVER UM POUCO MAIS."
110 END
120 FOR T=133 TO 1 STEP -5
130 PRINT "BANG!!!!!!!!!"
140 SOUND T, 1
150 NEXT T
160 CLS
170 PRINT @ 224, "SINTO MUITO, MAS VOCÊ ESTÁ MORTO"
180 SOUND 1,50
190 PRINT @ 290, "APRESENTE-SE A PRÓXIMA VÍTIMA"

```

### Explicações rápidas

1) É muito interessante aqui a presença da instrução END, caso em 10 tentativas você não se mate o TRS-2 manda-lhe uma mensagem e o programa termina.

Caso não houvesse a instrução END você que sobreviveu também ouviria o som do tiro que o matou????!!

Mas você está vivo ou morto?

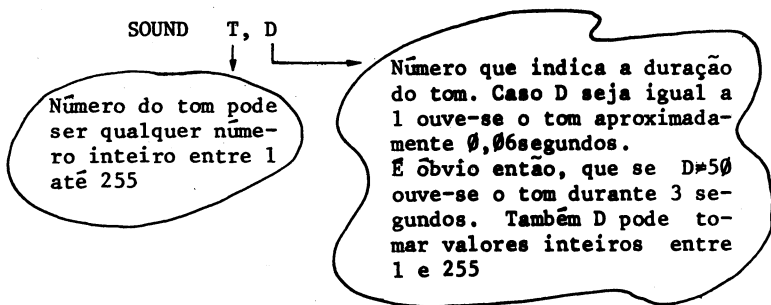
Resp.: Vivinho da silva se existir a instrução END.

Por outro lado se você der azar a linha 30 desvia o programa para a linha 120 onde é feita a sua execução, após o que surge a mensagem de pêsames e o TRS-2 para na linha 190 esperando a próxima vítima sem precisar da instrução END.

2) Finalmente vamos lhe apresentar a instrução  
SOUND

Para quem gosta de música esta instrução e o futuro domínio da instrução PLAY (abordamos muito o seu uso no livro TRS-2 Resolvendo os seus problemas") serão "ferramentas" suficientes para se obter maravilhas no campo sonoro.

A forma geral da instrução SOUND é



Com isto acredito que ficou claro o que significam as linhas 40 e 140.

Aliás mude a linha 40 para:

40 SOUND 200, 255 e concorde que o antigo "click" se transformou em uma pequena explosão!!!

# **CRAPS COM MÚSICA**

Quem já não ouviu falar do jogo do "craps" [crepe para os(as) íntimos(as)]?

No TK-Divertindo apresentamos uma versão deste jogo...

Para lhe mostrar mais um pouco do efeito sonoro que produz a instrução SOUND nada melhor do que elaborar o programa para o jogo do "craps" que permite a introdução de alguns alertas sonoros.

As regras do "craps", caso você as tenha esquecido, são as seguintes:

- 1) O TRS-2 lança para você dois dados.
- 2) Se você obtiver um dos seguintes resultados um tanto quanto raros logo de saída, soma de pontos S=2 ("olhos de cobra"), S=3 ou S=12 ("o máximo pioral") está frito(a) ou seja perde o jogo o qual obviamente termina aí.
- 3) Se você obtiver soma de pontos S=7 ou S=11 na primeira tentativa do TRS-2 ganhou a partida.  
Salte, pulule sem desmúnhocar...
- 4) Qualquer outra soma de pontos que for obtida no primeiro lançamento torna-se a "soma-meta". Ou será que é melhor falar "meta-soma"?  
Chi! Chi! Chi!

As duas coisas soam desagradavelmente e por isto passamos a chamar esta soma de "ponto ideal".

O TRS-2 joga os dados para você que torce desesperadamente para que saia o ponto ideal.

Bem aí vai o programa que é bastante auto explicativo.

```

10 CLS 8
20 D1 = RND(6): D2 = RND(6)
30 S = D1 + D2
40 PRINT @ 200, D1;
50 PRINT @ 218, D2;
60 PRINT @ 384, "VOCÊ OBTVE A SOMA □";S;
70 IF S=2 OR S=3 OR S=12 THEN 400
80 IF S=7 OR S=11 THEN 300
90 FOR K=1 TO 999: NEXT K
100 CLS 6
110 PRINT @ 192, "OBTENHA OUTRA SOMA □";S;
120 PRINT @ 224, "QUE AÍ VOCÊ VENCERÁ";
130 PRINT @ 288, "SE TIRAR UMA SOMA DE PONTOS 7";
140 PRINT @ 320, "O JOGO TERMINOU PARA VOCÊ";
150 PRINT @ 416, "APORTE <ENTER> QUANDO ESTIVER PRONTO PARA
    ACOMPANHAR O PRÓXIMO LANÇAMENTO";
160 R$ = CHR$(13): IF INKEY$ = R$ THEN 170 ELSE 160
170 X = RND(6): Y = RND(6)
180 Z = X+Y
190 CLS 2
200 PRINT @ 200, X;
210 PRINT @ 218, Y;
220 PRINT @ 384, "VOCÊ OBTVE A SOMA □";Z;
230 IF Z=S THEN 300
240 IF Z=7 THEN 400
250 FOR K=1 TO 999: NEXT K: CLS 4
260 GOTO 110

```

```

300 FOR K=1 TO 2999: NEXT K: CLS 2
310 FOR T=1 TO 255 STEP 5
320 SOUND T,1
330 NEXT T
340 FOR T=255 TO 1 STEP -5
350 SOUND T,1
360 NEXT T
370 PRINT @ 224, "PARABENS GANHADOR (GANHADORA)";
380 PRINT @ 288, "NÃO É EXAGERO DIZER QUE VOCÊ É MEIO
    RABUDINHO(A)";
390 GOTO 500
400 FOR K=1 TO 1999: NEXT K: CLS 7
410 FOR T=180 TO 210 STEP 5
420 SOUND T, 10
430 NEXT T
440 FOR J=1 TO 3
450 FOR T=200 TO 180 STEP -5
460 SOUND T, 20
470 NEXT T
480 NEXT J
490 PRINT @ 256, "SINTO MUITO MAS NÃO CONSEGUI TORNA-LO
    VITORIOSO(A)";
500 FOR K=1 TO 1999: NEXT K: CLS 5
510 PRINT @ 194, "QUE TAL UM NOVO JOGUINHO (S/N)?"
520 R$ = INKEY$: IF R$ = "S" THEN 10 ELSE 520

```

Execute e se divirta.

Depois da diversão descanse um pouco pois você precisa estar totalmente "aceso(a)" para o refinamento que irá aparecer a partir do Capítulo 9.



O.I. Caso você não tenha gostado da folha de papel apresentada na Figura 6 para trabalhar bem com a instrução PRINT @ e desta forma ter ra-



# O COSMO COLORIDO

É agora que vamos entrar de cheio no que se pode chamar "língua dos P".

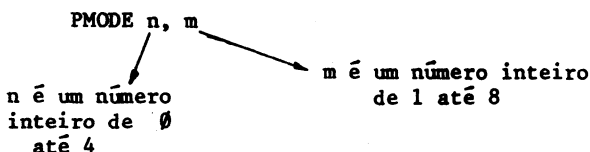
Ao comunicarmos as nossas vontades para o TRS-2 nesta forma a sua reação é de muita acurácia, muito detalhe e de muita beleza.

Para começar a sentir os astísticos recursos do seu TRS-2 digite o seguinte programa:

```
10 PMODE 4,1
20 PCLS
30 SCREEN 1,1
40 X=RND(164) + 50
50 Y = RND(100) + 50
60 Z = RND(41)
70 GOSUB 300
80 GOTO 40
300 FOR R=1 TO Z STEP 2
310 C = RND(4) + 1
320 CIRCLE (X,Y), R, C, .85
330 NEXT R
340 RETURN
```

*Explicações sobre o programa e as novas instruções*

Linha 10 → Aí aparece a instrução PMODE cuja forma geral é



O PMODE é a primeira instrução do CBE que você vai conhecer pois todas as outras que até agora utilizamos já existiam no COLOR BASIC.

Com o comando PMODE escolhe-se qual dos cinco modos gráficos será usado e para tanto basta definir o valor de n.

Com o n=0 ou n=1 escolhe-se o modo de *baixa resolução*.

Com o n=2 ou n=3 escolhe-se o modo de *média resolução*, e finalmente com n=4 escolhe-se o modo de *alta resolução* (veja a Tabela 4).

Instrução PMODE	Tamanho do gráfico	Número de cores	Tamanho relativo de um ponto na tela.
PMODE 4,1	256x192	2	□
PMODE 3,1	128x192	4	▢
PMODE 2,1	128x192	2	□
PMODE 1,1	128x96	4	▣
PMODE 0,1	128x96	2	▣

Tabela 4 - Modos gráficos

Com o inteiro m da instrução PMODE define-se a página de início ou melhor especifica-se em que página da memória o seu gráfico vai começar.



Como na linha 10 escolhemos PMODE 4,1 optamos pela alta resolução ou seja aquela que permite a maior quantidade de detalhes.

Neste modo o TRS-2 divide a tela da TV em 256 pontos sobre o eixo  $Ox$  e 192 pontos sobre o eixo  $Oy$  (veja a Figura 25).

Linha 20 → A forma geral desta instrução é

PCLS n

n pode ser qualquer expressão numérica cujo valor esteja entre 0 e 8.

Lembre que o n é opcional e quando é omitido é usada a cor do fundo

No caso do nosso programa o n não foi especificado e a cor de fundo atual é usada para limpar a tela de gráfico.

Resumindo com o PCLS limpa-se a tela de gráfico e é então uma instrução semelhante ao CLS que limpa a tela de texto.

Linha 30 → Aí temos uma outra instrução gráfica muito importante ou seja SCREEN 1,1 que é usada em combinação com a PMODE.

A sua forma geral é

SCREEN n, m

n é 0 (zero)  
ou 1 (um)

m é 0 (zero)  
ou 1 (um)

O inteiro n seleciona texto ou gráfico.

Se  $n=0$  escolhe-se a tela de texto e se  $n=1$  escolhe-se a tela para gráficos.

GRAPHICS SCREEN WORKSHEET (256 x 192)

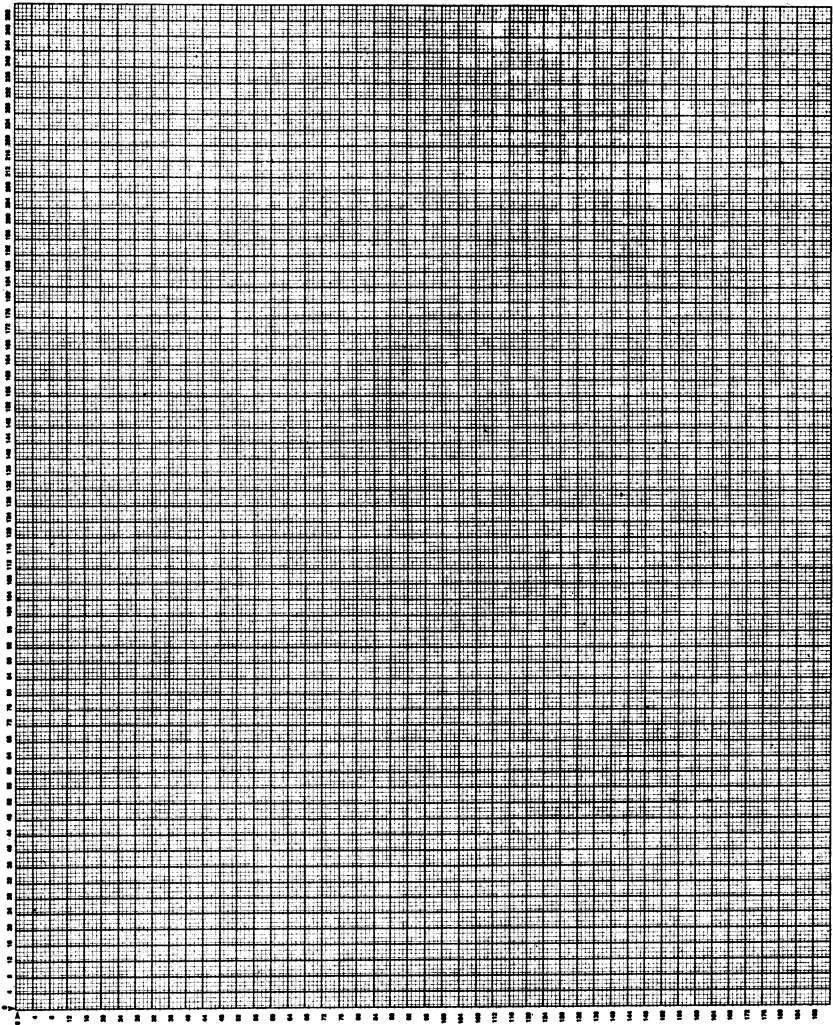


Figura 25 - Posições gráficas na tela  
no modo 4

O inteiro  $m$  escolhe um dos dois conjuntos de cores disponíveis para um particular modo gráfico chamado pelo PMODE.

Assim as duas instruções PMODE e SCREEN trabalham junto para produzir os formatos desejados.

Na Tabela 5 temos as possíveis combinações que podem ser usadas.

Instrução		Combinações de cores	
PMODE $n,1$ valores de $n$	SCREEN $1,m$ valores de $m$	duas cores	quatro cores
4	0	preto/verde	-
	1	preto/bege	
3	0		verde/amarelo/azul vermelho
	1		bege/ciano/lilás/ laranja
2	0	preto/verde	
	1	preto/bege	
1	0		verde/amarelo/azul vermelho
	1		bege/ciano/lilás/ laranja
0	0	preto/verde	
	1	preto/ bege	

Tabela 5 - Conjuntos de cores



0.I. Na Tabela 5 na combinação PMODE 4,1 e SCREEN 1,0 temos a cor do fundo preta e a cor do primeiro plano verde, já no caso da combinação PMODE 1,1 e SCREEN 1,1 temos uma cor de fundo

bege e as cores disponíveis para o primeiro plano são ciano, lilás ou laranja.

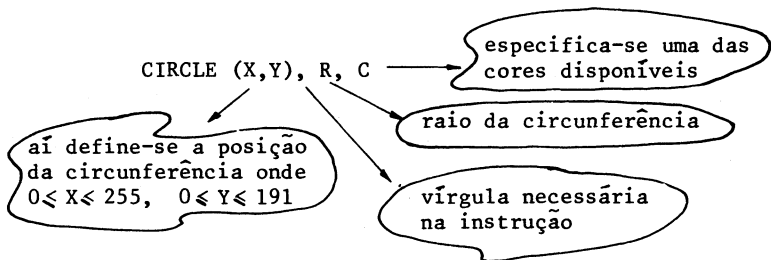
O mesmo tipo de interpretação se aplica a outras combinações de PMODE e SCREEN.

Linhas 40,50 e 60 → Sorteiam-se valores que serão atribuídos as variáveis X e Y, que definem o centro de uma circunferência, e a variável Z que servirá como valor limite do raio de uma circunferência que começará com um raio unitário e poderá chegar até o valor Z.

Linha 70 → Vai-se a uma subrotina que permite o desenho de várias circunferências.

Linha 320 → Está nesta linha uma das mais interessantes características do seu TRS-2 ou seja a possibilidade de desenhar uma circunferência de círculo, arcos de circunferência, circunferências achatadas parecendo elipses, etc. com uma única instrução.

A forma geral desta instrução é



Um raio R unitário é igual a um ponto ou seja uma posição da tela.

Caso o parâmetro C seja omitido, a cor do primeiro plano é usada para se desenhar a circunferência.

Porém na linha 320 temos algo mais depois do C ou seja o número 0.85.

Na realidade estamos na forma mais geral da instrução CIRCLE ou seja

CIRCLE (X,Y), R, C, E

excentricidade ou seja a relação entre altura e largura e pode ser um número entre 0 e 255

Com esta forma geral pode-se "comprimir" ou "esticar" uma circunferência como está mostrado na Figura 26.

Caso o parâmetro E seja omitido ele é tomado pelo próprio TRS-2 como sendo 1.

Finalmente se agora quisermos omitir o parâmetro para a cor mas ainda assim usar a excentricidade deve-se usar a instrução da seguinte forma:

CIRCLE (X,Y), R,, E

duas vírgulas para indicar que o parâmetro da cor foi omitido e a cor do primeiro plano é usada para a circunferência.

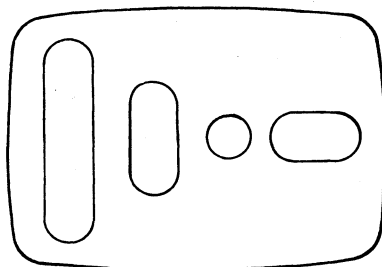


Figura 26 - Influência da excentricidade na instrução CIRCLE

Por enquanto sobre o CIRCLE é só isto, mas adiante tem mais...



```

70 CIRCLE (64,96), R, 2
60 COLOR 2,1
50 R = RND(30) + 30
40 FOR J=1 TO 13
30 SCREEN 1,0

```



O.I. Nós já falamos sobre isto mas não custa nada reforçar o conceito.

Na linha 90 tem-se uma linha de instruções múltiplas.

Assim em uma única linha temos

```

90 FOR K=1 TO 499: NEXT K

```

Diagram illustrating the structure of the instruction line 90: "FOR K=1 TO 499: NEXT K". A horizontal line is drawn under the text "FOR K=1 TO 499: NEXT K". Three arrows point downwards from this line to labels below: "instrução # 1" under "FOR", "dois pontos" under ":", and "instrução # 2" under "NEXT".

Para se ter uma linha de instruções múltiplas elas devem estar separados entre si obrigatoriamente por 2 pontos.

Pode-se ter uma linha com várias instruções (veja por exemplo a linha 300 do programa "BLACKJACK" do Capítulo 7).

Neste programa, o intuito da linha 90 de instruções múltiplas, como já foi em casos anteriores, é o de dar *uma pequena pausa*, para que você possa admirar as lindas circunferências que são *desenhadas*.

Em outras palavras, um pequeno intervalo de tempo entre um desenho e outro.





# POT-POURRI CIRCUNFERENCIAL

Vamos reforçar um pouco mais a importância do uso e da existência da instrução CIRCLE.

Espero que tudo o que se falou sobre PMODE, SCREEN, PCLS e COLOR não o(a) tenha deixado confuso(a) predisposto(a) leitor(a) a dominar o TRS-2

Mas, caso esteja um tanto obnubilado(a) volte ao Capítulo 9 e leia tudo de novo caso contrário siga em frente.

Aí está um lindo exemplo do IF-THEN e a decisão cabe a você.

Tecla agora o seguinte programinha para sentir o efeito da excentricidade.

```
10 PMODE 4,1
20 PCLS
30 SCREEN 1,0
40 CIRCLE (25,96), 18,, .4
50 CIRCLE (80,96), 18,, 1
60 CIRCLE (140,96), 18,, 2
70 CIRCLE (215,96), 18,, 5
80 GOTO 80
```

Execute e veja se sai algo semelhante ao que está indicado na Figura 26.

Note que nenhuma cor foi indicada (veja as duas vírgulas nas linhas 40 a 70) e portanto a cor verde do primeiro

plano é a que está sendo usada para obter os "contornos fechados".

Você já entendeu para que serve a excentricidade?

Ainda não!!!!

Bem, aí vai um programa com o qual desenha-se uma "circunferência" de raio 30 e cuja excentricidade vai sendo variada de 0 até 1,5 com incrementos de 0,1.

Além disto depois de um certo tempo que você ficou admirando a beleza que saiu na tela, manda-se uma instrução para apagar o que foi desenhado.

Acredito que com este programa também começará ficar mais evidente o que vem a ser a cor de primeiro plano e a cor do fundo.

```

10 PMODE 4,1
20 PCLS
30 SCREEN 1,0
40 FOR E=0 TO 1.5 STEP .1
50 CIRCLE (128,96),30,,E
60 FOR Z=1 TO 199: NEXT Z
70 CIRCLE (128,96),30,0,E
80 NEXT E
90 REM AGORA VAMOS COMEÇAR TUDO DE NOVO
100 GOTO 20

```

usa-se a cor do primeiro plano para desenhar

aí está uma pausa

usa-se a cor do fundo para apagar

Execute o programa.

Gostou do que viu?

Achou genial!?!?

Trilegal!!

Elimine a linha 70 do programa e com isto você não esquecerá mais o grande efeito de apagar obtido com o "zerinho" nesta instrução.

Você não esqueceu que para eliminar uma linha(a 70) basta você interromper o programa com um BREAK, teclar o nú-

merc da linha (70) e pressionar a tecla ENTER .

Para verificar que isto realmente aconteceu tecle LIST e aperte a tecla ENTER novamente.

Sumiu a linha 70?

Jóia, é o que queríamos!!!

Dê um RUN agora e veja como as coisas vão mudar muito.

Com o programa que vem imediatamente abaixo vamos desenhar 13 "circunferências" tendo os seguintes parâmetros aleatoriamente escolhidos.

- a) o centro da mesma;
- b) a sua cor;
- c) o seu raio;
- d) a sua excentricidade.

```

10 PMODE 3,1
20 PCLS 6
30 SCREEN 1,1
40 FOR J=1 TO 13
50 X=RND(120) + 64: Y = RND(48) + 48
60 R = RND(45)
70 C = RND(4) + 4
80 IF C = 6 THEN 70
90 E = RND(18)/10
100 CIRCLE (X,Y), R, C, E
110 NEXT J
120 GOTO 120

```

### *Explicações breves*

Linha 50 → É aí que se escolhem as coordenadas do centro.

Não esqueça que a maior circunferência que cabe na tela tem centro em (128,96) e um raio de 95 unidades.

Linha 70 → Pode-se ter um código de cor 5,6,7 ou 8.

Linha 80 → Aí impede-se que a circunferência tenha cor ciano (código 6) pois esta é a cor do fundo (veja a linha 20).

Quando C=6 volta-se a linha 70 para um novo sorteio.

Linha 90 → A excentricidade varia de 0,1 a 1,8.

Linha 120 → "Congelou-se" com esta instrução a maravilha que está na tela.

Caso você não queira nada estático pois não é um leitor(a) passivo(a) e muito ao contrário é todo(a) "agitadinho(a)" então mude a linha 120 e acrescente a 130, isto é:

```
120 FOR K=1 TO 1299: NEXT K
```

```
130 GOTO 20
```

Para terminar este capítulo vejamos uma mistura dos modos gráficos vendo um lindo tapete redondo colorido.

```
10 PMODE 3,1
```

```
20 PCLS
```

```
30 SCREEN 1,1
```

```
40 FOR I=1 TO 100
```

```
50 J = RND(4)
```

```
60 CIRCLE (128,96), I, J, .9
```

```
70 NEXT I
```

```
80 FOR P=1 TO 999: NEXT P
```

```
90 PMODE 4,1
```

```
100 SCREEN 1,1
```

```
110 FOR P=1 TO 999: NEXT P
```

```
120 PMODE 3,1
```

```
130 SCREEN 1,0
```

```
140 FOR P=1 TO 999: NEXT P
```

16Ø SCREEN 1,1

17Ø GOTO 8Ø

Vamos dar uma explicação mais detalhada do que ocorre com os pontos que aparecem na tela a medida que o programa vai mudando de PMODE.

Como foi indicado na Tabela 4 existem três modos quanto ao tamanho do ponto que aparece na tela.

Assim no modo de resolução 4 o ponto 137,95 aparece como indicado na Figura 27a.

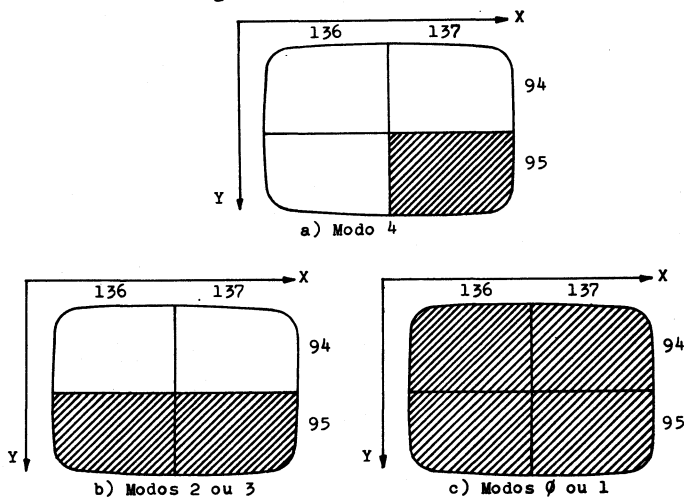


Figura 27

Já no modo de média resolução (2 ou 3) os pontos (136,95) e (137,95) apareceram na tela embora se especifique apenas (137,95) conforme mostrado na Figura 27b.

No modo de baixa resolução (Ø ou 1) os pontos (136,94); (137,94); (136,95) e (137,95) aparecerão na tela embora o ponto especificado tenha sido apenas o (137,95) conforme mostrado na Figura 27c.

Desta forma o modo de baixa resolução faz aparecer 4 pontos (ou pixels), o modo de média resolução faz surgir 2

GRAPHICS SCREEN WORKSHEET (128 x 192)

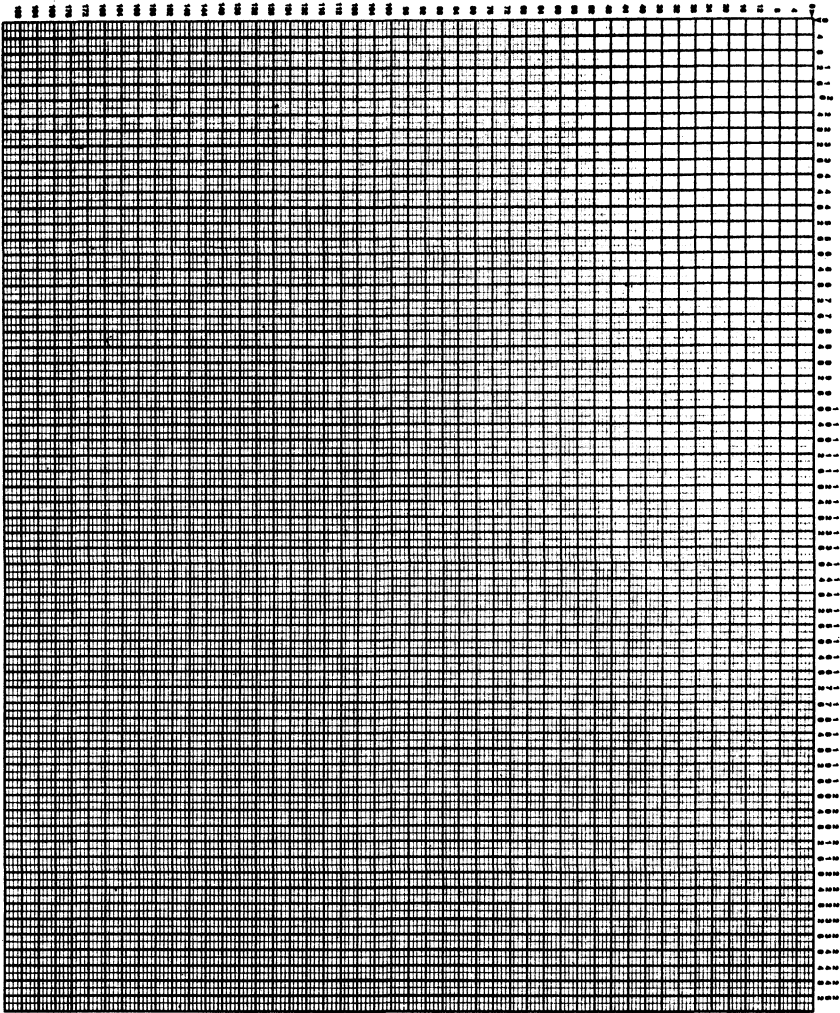


Figura 28a - Modos 2 e 3

GRAPHICS SCREEN WORKSHEET (128 x 96)

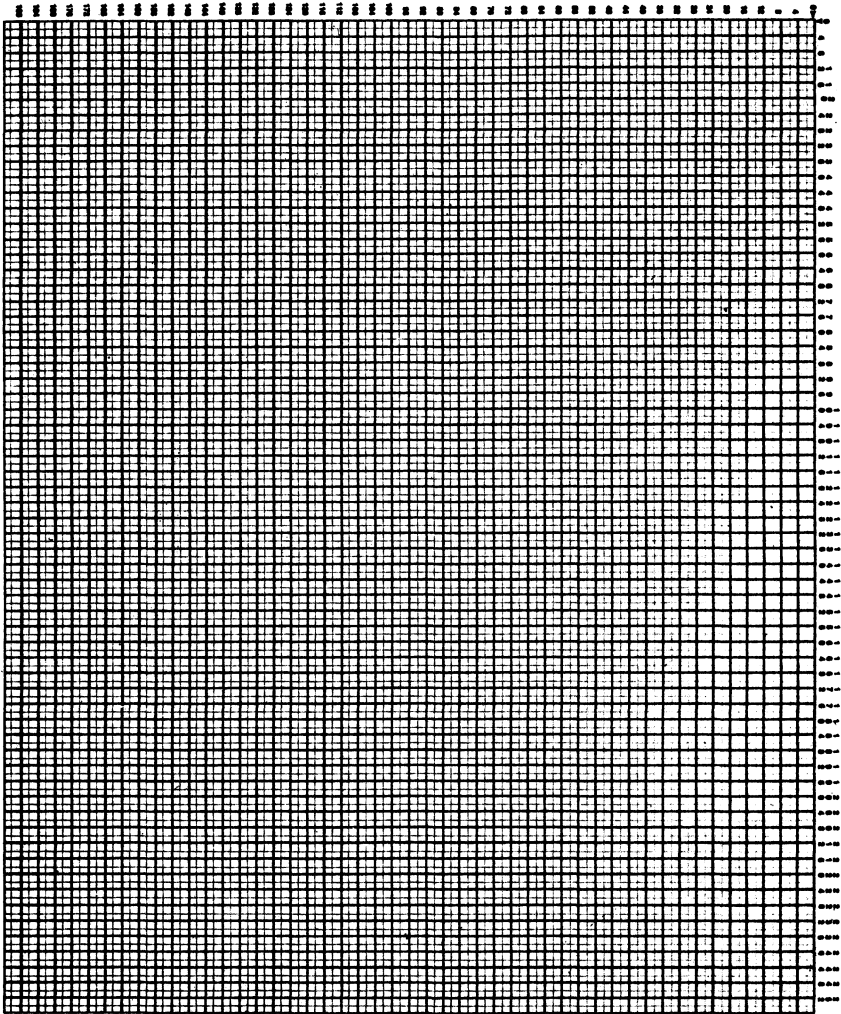


Figura 28b - Modos 0 e 1

pontos e no modo de alta resolução surge apenas um ponto.

Especificando-se qualquer um dos pontos (136,94); (137,94); (136,95) ou (137,95) no modo  $\emptyset$  trará como resultado na tela os quatro pontos.

Especificando-se os pontos (136,95) ou (137,95) nos modos 2 ou 3 trará como resultado na tela os dois pontos.

No modo 4 o ponto especificado é o único ponto que aparece na tela.

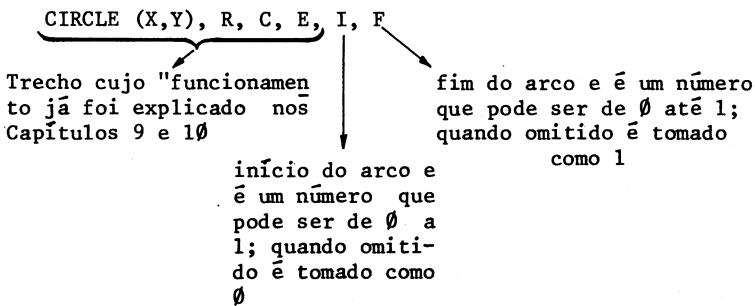
Nas Figuras 28a e 28b apresentamos o quadriculado da tela respectivamente para as posições na tela os modos 3 e 2 e as posições na tela nos modos  $\emptyset$  e 1.



# DESENHANDO ARCOS

A versátil instrução CIRCLE pode ter mais dois parâmetros com os quais pode-se desenhar apenas partes da circunferência ou seja arcos.

A forma geral completa da CIRCLE é:



O que significam estes valores de 0 a 1 que podem ser atribuídos a I e F?

Resp.: Para localizar a posição 0 (zero) pense nos "antigos" relógios na posição 3 horas (veja a Figura 29a).

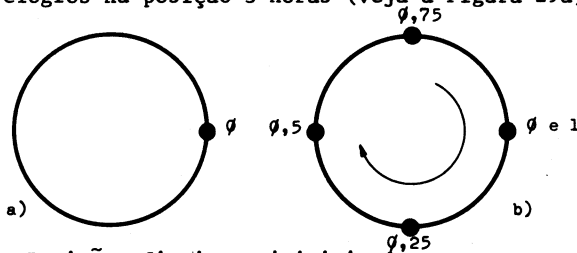


Figura 29 - Posições finais ou iniciais de um arco



O.I. Espero que o uso da vírgula decimal, assim como aparece nos números escritos junto a circunferência na Figura 29b ou do ponto decimal, como exige o C B E ao se escrever os números, nunca lhe traga alguma interpretação errônea.

As outras posições entre 0 e 1, no sentido horário estão na Figura 29b.

Nestas condições para se obter um arco de circunferência amarelo centrado no meio da tela (Figura 30) basta escrever: CIRCLE (128,96),40,2,1,6,.3

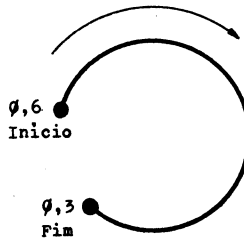


Figura 30



O.I. Para se usar os pontos inicial (I) e final (F) com o intuito de se desenhar um arco, você precisa especificar a excentricidade (a razão altura/largura) que para uma circunferência normal no seu TRS-2 é tomada como valendo 1.

Vamos agora elaborar um programa que permita apresentar na tela um conjunto de arcos como mostrado na Figura 31.

```
10 PMODE 4,1
20 PCLS
30 SCREEN 1,0
40 I = .3 : F = .7
50 FOR R = 40 TO 60 STEP 5
```

```

60 CIRCLE(128,96), R,,1,I,F → arcos da direita da Fig. 31.
70 CIRCLE(128,96),R,,1,.5+I,.5-I → arcos da esquerda da
    Figura 31
80 NEXT R
90 GOTO 90 → vamos "congelar" esta maravilha

```

Se "você quer um pouco de som introduza as linhas

```

65 FOR T = 200 TO 203: SOUND T,1: NEXT T
75 FOR T = 1 TO 4: SOUND T,1: NEXT T

```

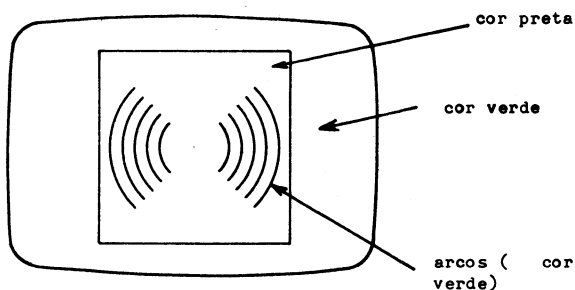


Figura 31

Que tal, você agora elaborar um programa que permita obter um desenho como o mostrado na Figura 32?

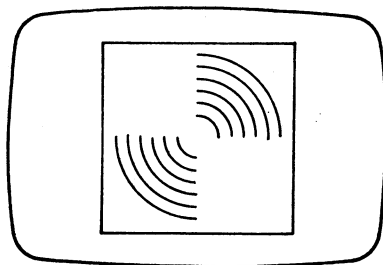


Figura 32

Vamos agora sofisticar e tudo isto é possível em vista destes dois últimos parâmetros que a instrução CIRCLE possui.

O que você vai ver daqui a pouco na tela do TRS-2 é o "redondão" da Figura 33.

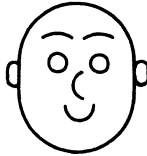


Figura 33

Aí vai o belo programa "REDONDÃO"

```

10 PMODE 4,1
20 PCLS
30 SCREEN 1,1
40 CIRCLE(128,96),52,,1.22 → instrução para se ter a cabeça
50 CIRCLE (110,79),4 } → instruções para se ter os olhos
60 CIRCLE (136,79),4 }
70 CIRCLE (120,100),6,,1.3,.14,.76 → instrução para se ter o nariz
80 CIRCLE (110,74),8,,1,.64,.86 } → instruções para obter as
90 CIRCLE (136,74),8,,1,.64,.86 } sobancelhas
100 CIRCLE(120,121),11,,1,.09,.41 → instrução para se obter a boca
110 CIRCLE(70,96),8,,1.85,0,.9 } → instruções para se obter as orelhas
120 CIRCLE(186,96),8,,1.85,.55,.45 }
130 LINE (20,20)-(230,178), PSET, B → uma moldura para o lindo "redondão" pode ser obtido com esta instrução. (veja o Capítulo 14).
140 GOTO 140

```

# TRACANDO LINHAS RETAS COLORIDAS

Propositadamente invertemos a ordem natural das coisas e ensinamos a você a partir do Capítulo 9 como mexer com curvas e agora vamos nos enturmar com as diversas formas de elaborar as linhas retas.

Para começar tecle o seguinte programa:

```

10 PMODE 4,1 → aí escolhe-se o modo gráfico
20 PCLS → limpa-se a tela com a cor do fundo
30 SCREEN 1,1 → cor do fundo preta e cor do primeiro pla
no bege
40 FOR X=120 TO 150
50 PSET (X,25,5)
60 NEXT X
70 GOTO 70

```

└──────────→ cor bege (ou talvez cinza  
no seu aparelho de TV)

Na linha 50 aparece uma instrução nova ou seja PSET  
A forma geral desta instrução é:

PSET(X,Y,C)

especifica a cor do ponto e pode ser um valor (expressão numérica) entre 0 e 8.

especifica uma posição sobre o eixo Ox (horizontal) e pode ser uma expressão numérica cujo valor deve estar entre 0 e 255

especifica uma posição sobre o eixo Oy (vertical) e pode ser uma expressão numérica cujo valor deve estar entre 0 e 191.

Portanto no nosso caso particular na linha 50 indica-se que um ponto bege é colocado na posição (X,25) e como X vai variar de 120 a 150 de um em um, isto vai fazer com que apareça um segmento retilíneo como mostrado na Figura 34.

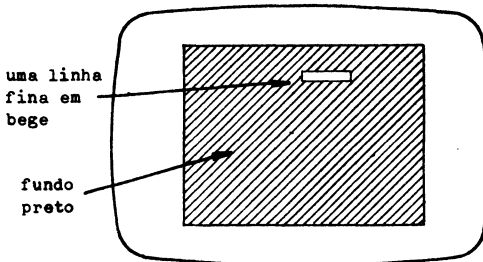


Figura 34

Caso você queira modificar as cores do fundo e do primeiro plano basta colocar a instrução

```
15 COLOR 0,5
      |  ↘
      |   fundo bege (cinza)
      |
      |   primeiro plano preto
```

É evidente que também se deve modificar a linha 50 para:

```
50 PSET (X,25,0)
      |  ↘
      |   preta é agora a cor
      |   do primeiro plano
```

Uma última mudança que convém você fazer no programa seria perceber mais uma vez a diferença na largura da linha quando se passa para os modos 2 e 0.

Altere então a linha 10 inicialmente para

```
10 PMODE 2,1 e depois para 10 PMODE 0,1
```

Dá para notar que com o PMODE 0,1 a linha desenhada é bem mais larga?

Uuoooootimo!!!!

E aí vai uma aplicação mais interessante da instrução PSET.

Que tal você ver na tela uma moldura como a mostrada na Figura 35?

Gostou da idéia?

Então vamos ao programa "MOLDURA MULTICOLORIDA"

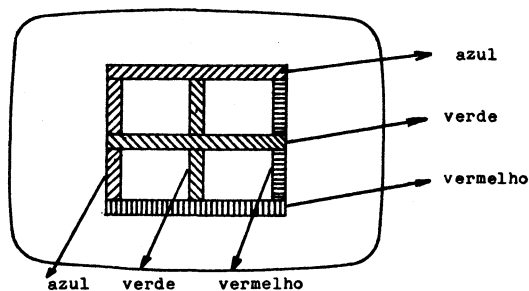


Figura 35

```

10 PMODE 1,1
20 COLOR 1,2 —————> fundo amarelo
30 PCLS —————> primeiro plano verde
40 SCREEN 1,0
50 REM VAMOS DESENHAR AS LINHAS VERTICAIS
60 FOR Y=60 TO 120
70 PSET (72,Y,3)
80 PSET (107,Y,1)
90 PSET (142,Y,4)
100 NEXT Y
110 REM VAMOS AGORA DESENHAR AS LINHAS HORIZONTAIS
120 FOR X=72 TO 142
130 PSET (X,60,3)
140 PSET (X,90,1)
150 PSET (X,120,4)
160 NEXT X

```

170 REM CHEGOU A HORA DE FICAR PASMADO COM O QUE ESTÁ NA TELA

180 GOTO 180 —————> tela "congelada"

Dê um RUN e admire!!!

Digamos que você queira apagar alguma linha.

Neste caso deve ser usada a instrução PRESET.

A sua forma geral é

PRESET (X,Y)

X especifica a posição sobre o eixo Ox e pode ser uma expressão numérica com valor entre 0 e 255.

Y especifica a posição sobre o eixo Oy e pode ser uma expressão numérica com valor entre 0 e 191.

Você vai ver que usando esta instrução é tão fácil apagar um "pingo" ou seja uma posição, como ativar a mesma.

Lembra-se ainda que o RESET tinha esta função quando se queria apagar o que foi ativado pelo SET?

Bom, nota dez para você!!!

Aproveite o programa "MOLDURA MULTICOLORIDA" e adicione as seguintes linhas para ver o efeito do PRESET.

```
180 REM VAMOS APAGAR AS LINHAS VERDES INTERNAS
190 FOR X=74 TO 140
200 PRESET (X,90)
205 NEXT X
210 FOR Y=60 TO 120
220 PRESET (107,Y)
230 NEXT Y
240 REM AÍ VAI UMA PAUSA SEM FIM
250 GOTO 250
```

Execute e verifique se o preconizado ocorreu...



# TOMANDO A SUA LINHA

Você já sabe como ativar pontos e apagar os mesmos em qualquer ponto da tela na "língua dos P".

Porém o seu C B E tem uma instrução geral que permite com grande facilidade e muito menos trabalho do que aquele feito no Capítulo 12 obter a representação de retas.

Estou me referindo a instrução LINE cuja forma geral é:

LINE  $(X_1, Y_1) - (X_2, Y_2)$ , PSET  $\longrightarrow$  ativa todos os pontos que unem em linha reta (menor distância entre dois pontos no plano) o ponto inicial e o final

ponto inicial (coluna, linha)      ponto final (coluna, linha)

Em primeiro lugar vai um programa que produz quatro retas (uma azul, uma amarela, uma verde e uma vermelha) separadas entre si por algumas linhas (Figura 36).

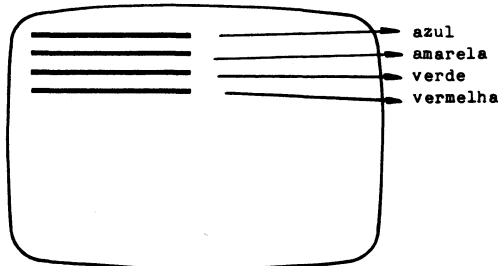


Figura 36

```

10 PMODE 1,1
20 PCLS
30 COLOR 3,1
40 SCREEN 1,0
50 REM AI VÃO AS LINHAS RETAS COLORIDAS
60 LINE(0,10) - (140,10), PSET → 1ª linha reta em azul
70 COLOR 2,1
80 LINE(0,25) - (140,25), PSET → 2ª linha reta em amarelo
90 COLOR 1,1
100 LINE(0,40) - (140,40), PSET → 3ª linha reta em verde
110 COLOR 4,1
120 LINE(0,55) - (140,55), PSET → 4ª linha reta em vermelho
130 GOTO 130

```

Execute e veja se sai o que está indicado na Figura 36.

Vamos agora fazer algumas sutis mudanças e em cada uma delas você verá acontecer o seguinte:

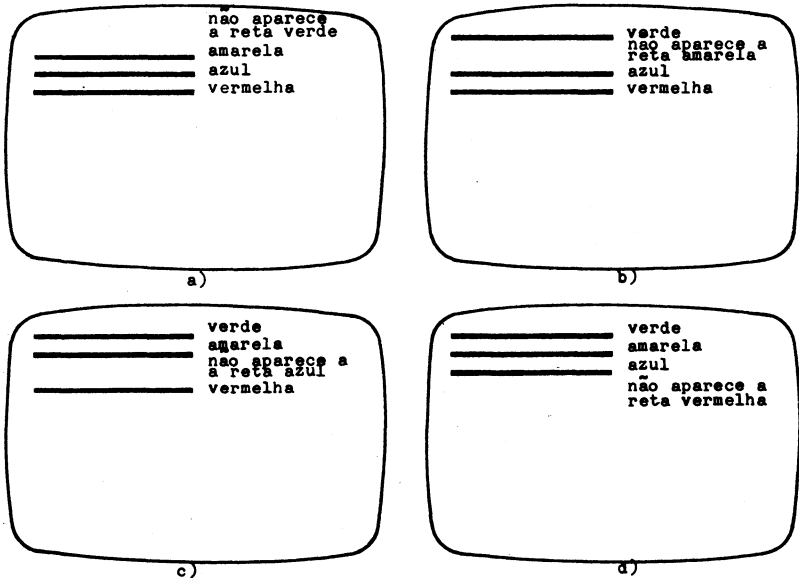


Figura 37

Aí vai o programa:

```

10 REM ESCOLHENDO O MODO E A COR
20 PMODE 1,1
30 FOR FUNDO = 1 TO 4
40 PCLS FUNDO
50 SCREEN 1,0
60 REM E A PARTIR DESTA PONTO QUE SE DESENHAM AS RETAS
70 FOR C=1 TO 4
80 Y = 10 * C
85 COLOR C, FUNDO
90 LINE (0,Y) - (140,Y), PSET
100 NEXT C
110 FOR K=1 TO 399 : NEXT K → pausa para uma contemplação
120 NEXT FUNDO
130 GOTO 30 → vai começar tudo de novo

```

Parece que você já está apto a desenhar linhas horizontais!?!?

Para desenhar as verticais é a mesma coisa, porém em caso de dúvida não esqueça de que os quatro cantos da tela no PMODE são:

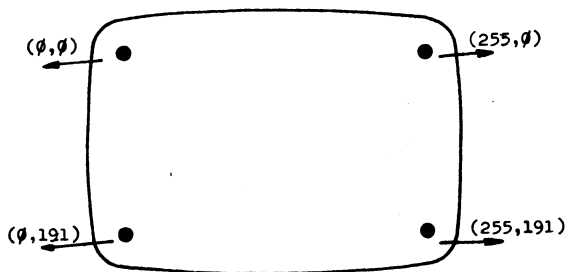


Figura 38

Que tal traçar diagonais unindo os pontos extremos indicados na Figura 38, além de uma vertical passando pelos pontos  $(250,0)$  e  $(250,191)$ .

Gostou da idéia, não é?  
 Tecle então:

```

10 REM  DIAGONAIS E UMA VERTICAL
20 PMODE 0,1
30 PCLS
40 SCREEN 1,1
50 LINE (0,191) - (255,0), PSET }
60 LINE (0,0) - (255,191), PSET }  estas são retas na
                                   diagonal
70 LINE (250,0) - (250,191), PSET  →  reta vertical da di-
                                   reita

80 REM  AÍ VAI UM "SONZINHO"
90 FOR T=58 TO 79 STEP 2: SOUND T,6: NEXT T
100 GOTO 100

```

*Sugestão:* Se gostou do que saiu e quer tentar outras combinações de cores faça as seguintes possíveis mudanças.

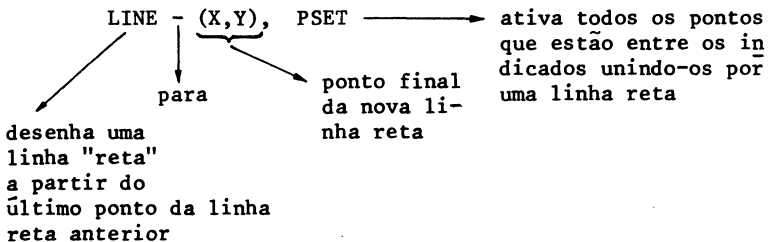
- |               |               |
|---------------|---------------|
| a) 30 PCLS 5  | b) 30 PCLS 1  |
| 35 COLOR 0,5  | 35 COLOR 0,1  |
| 40 SCREEN 1,1 | 40 SCREEN 1,0 |

Agora que você já viu como se traçam as retas horizontais, verticais e na diagonal uma idéia interessante é aquela de formar figuras geométricas (retângulos, quadrados, losangos, triângulos, etc).

Para isto basta unir as retas.

O TRS-2 permite que se use a instrução LINE de uma forma diferente quando se está desenhando linhas retas que se quer unir.

A forma geral da instrução LINE para esta finalidade é:



Para mostrar o efeito dessa instrução vamos desenhar um triângulo.

```

10 PMODE 1,1
20 PCLS
30 COLOR 3,1
40 SCREEN 1,0
50 LINE (78,10) - (168,10), PSET
60 LINE - (110,70), PSET
70 LINE - (78,10), PSET
80 GOTO 80

```

Execute e veja se sai na tela um triângulo.

Será que você é capaz agora de obter o desenho da Figura 39?

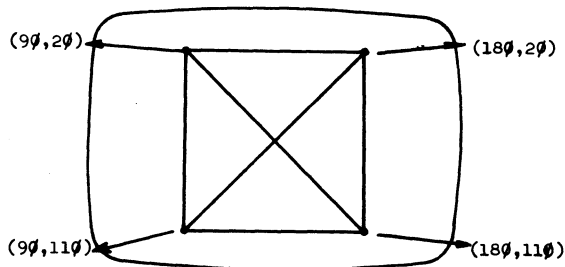


Figura 39

Você pode usar qualquer modo gráfico e as cores que mais lhe apatecerem.

O que, você tem dificuldade é com o programa e não com as cores?

É, isto até que é compreensível e por isto vou fazer o programa para o(a) digno(a) leitor(a).

```

10 PMODE 4,1
20 PCLS
30 SCREEN 1,0
40 LINE (90,20) - (180,20), PSET
50 LINE - (180,110), PSET
60 LINE - (90,20), PSET
70 LINE - (90,110), PSET
80 LINE - (180,110), PSET
90 LINE - (180,20), PSET
100 LINE - (90,110), PSET
110 GOTO 110

```

Explicação geométrica do programa

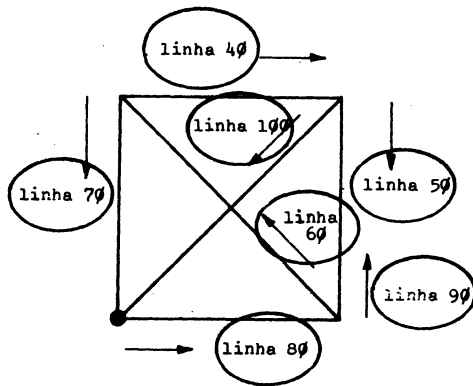


Figura 40

# 14 MOLDURAS VAZIAS E COLORIDAS

Já deu para perceber que com a instrução LINE se pode desenhar por exemplo contornos retangulares bem mais rapidamente do que ativando os pontos individualmente com o PSET.

Existe inclusive no seu TRS-2 uma possibilidade de desenhar um contorno retangular (moldura) usando um parâmetro adicional dentro da instrução LINE.

Pense em uma moldura ou uma caixa ("box" em inglês) cujas coordenadas para os quatro cantos estão indicadas na Figura 41.

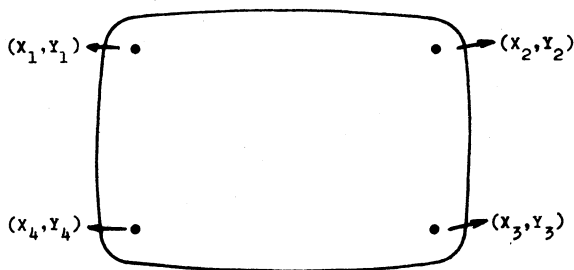


Figura 41

Deve-se usar a instrução

LINE  $(X_1, Y_1) - (X_3, Y_3)$ , PSET, B → faz desenhar a moldura retangular

um canto ou vértice      vértice ou canto oposto      ativa todos os pontos

Os pontos  $(X_2, Y_2)$  e  $(X_4, Y_4)$  podem também ser usados no lugar de  $(X_1, Y_1)$  e  $(X_3, Y_3)$  pois representam cantos opostos.

Para que você se convença disto tecle o seguinte pequeno mas importante programa.

```
10 PMODE 3,1
20 PCLS
30 SCREEN 1,0
40 LINE (40,80) - (120,160), PSET, B
50 GOTO 50
```

A linha 40 pode também ser:

```
40 LINE (120,80) - (40,160), PSET, B
```

ou

```
40 LINE (120,160) - (40,80), PSET, B
```

ou ainda

```
40 LINE (40,160) - (120,80), PSET, B
```

Tente todas estas possibilidades e uma vez constatada a veracidade não tem mais o que falar de como surgiu a moldura em torno do "REDONDÃO" no Capítulo 11.

Vamos agora apresentar-lhe "quadrado(a)" leitor (leitora), com todo respeito, um programa que enche a tela com 13 molduras retangulares colocadas aleatoriamente na tela.

```
10 PMODE 3,1
20 PCLS
30 SCREEN 1,0
40 FOR V=1 TO 13
50 Y = RND(60): X = RND(115)
60 LINE (X,Y) - (X+Y, X+Y), PSET, B
70 NEXT V
80 GOTO 80
```



Execute o programa várias vezes e veja que lindos "quadros" você obtém!!!

Caso você queira ver tudo multicolorido acrescente as seguintes linhas.

```
42 C = RND(4)
```

```
44 COLOR C,1
```

```
46 IF C=1 THEN 44
```

como as vezes pode ser escolhida a cor verde e o fundo é verde então esta "moldura" você não verá e com esta instrução elimina-se esta possibilidade

Uma outra mudança interessante é aquela ligada ao PMODE, PCLS, C e COLOR para produzir uma variedade maior de padrões de cores na tela.

Já que estamos falando de cores, que como já deu para perceber é o assunto básico deste livro que tal preencher o que está dentro do contorno retangular com alguma cor ou o que é mesma coisa pintar o mesmo?

É uma idéia bastante interessante, não é?

Para tanto vamos apresentar uma outra opção de aplicação da instrução LINE

LINE  $(X_1, Y_1) - (X_3, Y_3)$ , PSET, BF

Esta parte você já sabe o que significa

em inglês "fill the box" significa encher a caixa ou seja o contorno retangular

Para sentir a força deste F após o B tecler o seguinte programinha que permite apresentar um retângulo laranja, dois em cor lilás e um na cor ciano conforme se mostra na Fig. 42.

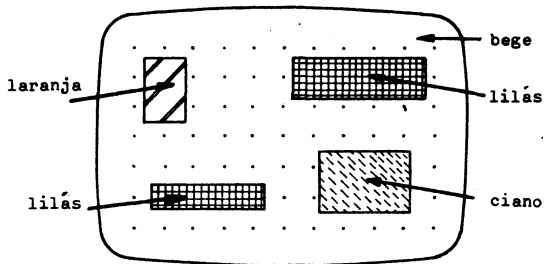


Figura 42

```

1Ø PMODE 1,1
2Ø PCLS
3Ø SCREEN 1,1
4Ø COLOR 8,5
5Ø LINE (27,19)-(42,76), PSET, BF → retângulo laranja
6Ø COLOR 7,5
7Ø LINE (22,145)-(95,158), PSET, BF } → retângulos lilás
8Ø LINE (148,55)-(207,83), PSET, BF }
9Ø COLOR 6,5
10Ø LINE (131,128)-(215,163), PSET, BF → retângulo ciano
11Ø GOTO 11Ø

```

Execute e pasme com o efeito do BF.

Aproveitemos o programa que colocava 13 retângulos aleatoriamente na tela e após algumas mudanças pintemos os mesmos

```

1Ø PMODE 3,1
2Ø PCLS
3Ø SCREEN 1,1
4Ø FOR V=1 TO 13
5Ø C = RND(4) + 4
6Ø IF C=5 THEN 5Ø → impede-se a escolha da cor do fundo
7Ø COLOR C,5 → bege (ou cinza)

```

```

30 X=RND(115): Y=RND(60)
90 LINE(X,Y)-(X+Y ,X+Y), PSET, BF → com isto preenche-se
100 NEXT V                               todo o retângulo com
110 GOTO 110                               a última cor de pri-
                                           meiro plano

```

Note que alguns dos retângulos coloridos podem superpor-se a outros e com isto eles apagam parte ou totalmente outros retângulos.

Aí vai um pequeno probleminha para você dedicado(a) leitor(leitora).

Elabore o programa que permita obter os desenhos das Figuras 43 a), b) e c).

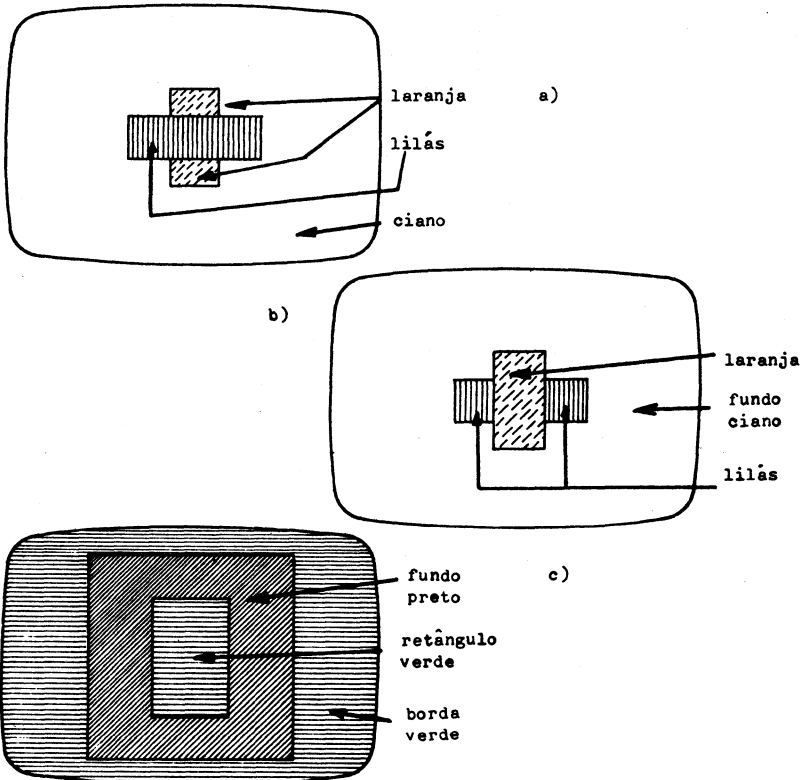
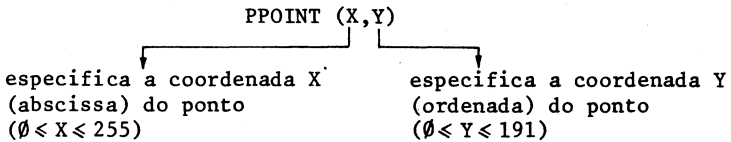


Figura 43



# ACERTANDO NO PUNTO

Uma última função P que lhe pode ser necessária é a PPOINT cuja forma geral é



Com PPOINT pode-se testar se um ponto gráfico especificado está aceso (ativado) ou apagado e além disto obtém-se o código da cor do ponto testado.

Suponha que você coloriu uma moldura retangular com cor vermelha e o fundo é verde e está querendo testar isto.

Como exemplo acompanhe o seguinte programinha

```

10 PMODE 3,1
20 PCLS
30 SCREEN 1,0
40 LINE (30,100) - (130,150), PSET, BF
50 FOR K=1 TO 999: NEXT K
60 PRINT PPOINT(50,125)
70 PRINT PPOINT (150,18)
  
```

*Explicações*

Linha 60 → Em vista desta instrução o TRS-2 lhe imprimirá 4 pois o ponto (50,125) está dentro da área vermelha.

Linha 70 → Em vista desta instrução o TRS-2 lhe imprimirá 1 pois o ponto (150,18) está na região verde.

Você percebeu que quando as instruções de impressão PRINT das linhas 60 e 70 são executadas, o TRS-2 retorna ao modo texto para mostrar os resultados.

Isto é feito automaticamente quando a instrução PRINT é executada independente do modo atual que se esteja.

Em outras palavras, toda vez que o seu programa pede texto com o PRINT (o mesmo ocorre com o INPUT) o TRS-2 realiza automaticamente um comando SCREEN 0,0.

Agora vamos elaborar um programa que fará com que metade da tela seja laranja e a outra metade de cor ciano..

O seu TRS-2 irá testar realmente se a cor é esta e depois de uma pequena pausa exibirá novamente a tela com duas cores mostrando-lhe com isto que quando se volta do modo texto não quer dizer que sumiu da memória do TRS-2 a parte gráfica.

Saiba que os dados gráficos estão guardados em uma área especial da memória e que não é a mesma onde está a parte de texto.

Bem, aí vai o programa

```
10 PMODE 3,1
20 PCLS
30 SCREEN 1,1
40 LINE (0,0)-(255,95), PSET, BF
50 COLOR 6,8
60 LINE (0,96)-(255,191), PSET, BF
70 FOR K=1 TO 450: NEXT K
```

```

80 CLS
90 REM AGORA VAMOS PEDIR PARA IMPRIMIR A COR DOS PONTOS
100 PRINT PPOINT (75,40)
110 PRINT PPOINT (145,138)
120 FOR K=1 TO 450: NEXT K
130 SCREEN 1,1
140 FOR K=1 TO 450: NEXT K
150 GOTO 80

```

Execute e veja se sai o que está indicado na Figura 44.

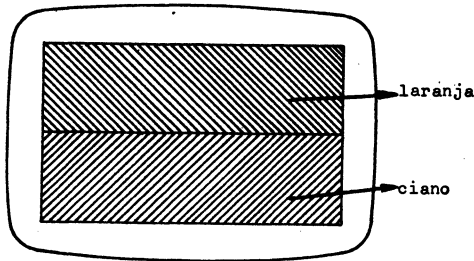


Figura 44

#### *Explicações*

Linha 100 → Devido a esta instrução sai impresso 8

Linha 110 → Devido a esta instrução sai impresso 6

Como uma aplicação um pouco mais importante de PPOINT aí vai um programa com o qual se apresentam 28 círculos com a cor escolhida aleatoriamente e posicionados também aleatoriamente na tela.

Você provavelmente quererá ver todas as 28 circunferências e é aí que entra a instrução PPOINT para testar a cor do ponto onde uma nova circunferência é desenhada.

Se este ponto tem a mesma cor que a escolhida para uma nova circunferência o TRS-2 apaga a circunferência antiga e não desenha uma nova.

O programa é o seguinte:

```

10 PMODE 3,1 → ajusta-se a tela.
20 PCLS
30 SCREEN 1,0.
40 FOR J=1 TO 28 → o procedimento será repetido 28 vezes
50 X = RND(210) + 20
60 Y = RND(150) + 20
70 IF PPOINT (X,Y) = C THEN PAINT (X,Y), 1,1: GOTO 120
    ↓
    → testa-se o ponto
80 C = RND(4)
90 IF C=1 THEN 80 → aqui rejeita-se a cor verde
100 CIRCLE (X,Y), 13, C → desenha-se uma nova circunferência
110 PAINT (X,Y), C, C → pinta-se a parte interna da circunferência obtendo-se um círculo
120 NEXT J
130 FOR K=1 TO 799: NEXT K
140 GOTO 20 → volta-se para começar tudo de novo

```

Antes de executar o programa leia as seguintes explicações.

Linha 50 e 60 → Aí escolhem-se as coordenadas do centro da circunferência ( $20 < X < 230$  e  $20 < Y < 170$ )

Linha 70 → O ponto escolhido para ser o centro é testado quanto a cor

Se a cor C que será usada já existe no ponto X,Y com esta instrução pinta-se esta área de verde (cor do fundo) de forma que ela será apagada e uma nova circunferência não é desenhada neste caso.

Mas quem explicou como funciona a instrução PAINT?

Resp.: Até agora ninguém, isto é só uma menção antecipada antes da explicação detalhada sobre como funciona PAINT que é dada no Capítulo 16.



A linha 70 é uma linha com instruções múltiplas e se no teste a condição é verdadeira vai-se para a instrução GOTO que desvia o programa para a instrução 120 impedindo com isto, como já dissemos, o desenho da circunferência.

Caso uma nova circunferência tenha que ser desenhada isto quer dizer que na instrução IF ocorreu a conclusão falso e aí sim passa-se para a linha 80.

Linhas 80 a 110 → Aí escolhe-se uma cor para a nova circunferência entre amarelo, azul ou vermelho.

A cor verde é rejeitada na linha 90.

As circunferências são desenhadas na linha 100 e pintadas com a mesma cor na linha 110.

O procedimento é repetido 28 vezes devido ao laço FOR-NEXT.

Na tela provavelmente não aparecerão 28 círculos devido ao fato de que alguns serão apagados ao se passar pela linha 70.

Linha 140 → Volta-se praticamente ao início do programa para começar tudo de novo.

Bem agora você pode executar o programa e deseje-lhe muita sorte para que possa ver as 28 circunferências diferentes quando aí então pode parar de olhar.

Vou dar um desconto de 25%.

Contou 21 circunferências pode dar um BREAK.

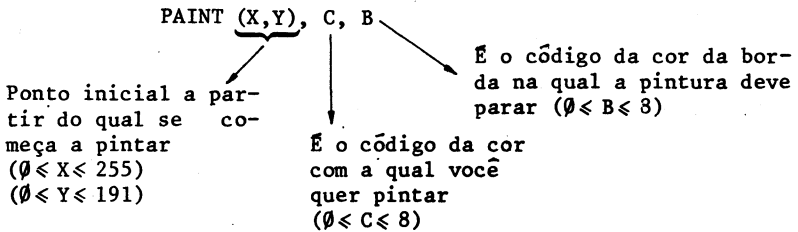


# PINTANDO O "SETE"

A função gráfica PAINT irá possibilitar a você a opção de poder aposentar os pinceis, os lápis coloridos, as tintas, etc.

Há pouco já usamos o PAINT (Capítulo 15) mas é agora que você vai ver como se pinta com o PAINT as diferentes figuras geométricas.

A forma geral da instrução PAINT é



Caso o TRS-2 atinja as cores da borda que diferem da cor da borda especificada ele continua pintando além desta borda.



Figura 45

Para começar vamos fazer um programa no qual temos duas diagonais máximas cruzadas na tela e uma circunferência com centro no centro da tela e de raio 65 unidades e a seguir toma-se um ponto em um dos quatro setores (veja a Figura 46) em que se dividiu o círculo e a partir daí pinta-se o setor todo.

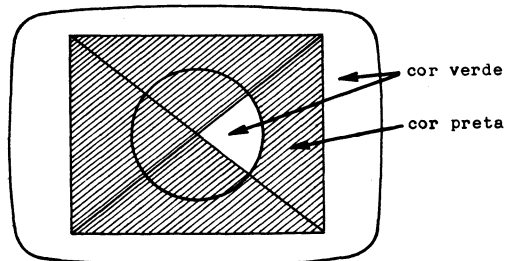


Figura 46

```

10 PMODE 4,1
20 PCLS
30 SCREEN 1,0
40 LINE (0,0) - (255,191), PSET
50 LINE (255,0) - (0,191), PSET
60 CIRCLE (128,96), 65
70 PAINT (158,105), 1,1
80 GOTO 80

```

Vamos agora a um programa que pinta círculos concêntricos

```

10 REM CÍRCULOS COLORIDOS
20 PMODE 3,1
30 PCLS
40 SCREEN 1,1
50 FOR R=30 TO 72 STEP 4 → aqui se varia o raio de
                        cada circunferência
60 C = RND(4) + 4
70 COLOR C,5
80 CIRCLE (128,96), R,C
90 NEXT R
100 C = RND(4) + 4
110 B = RND(4) + 4
120 PAINT (128,96), C, B
130 FOR K=1 TO 399: NEXT K
140 GOTO 50

```

Execute e quando você cansar de tanta pintura circular dê um BREAK e passe para o programa seguinte

```

10 REM QUADRADOS COLORIDOS
20 PMODE 3,1
30 PCLS
40 SCREEN 1,1
50 FOR L=30 TO 85 STEP 5
60 C = RND(4) + 4
70 COLOR C,5
80 LINE (128-L, 96-L) - (128+L, 96+L), PSET, B
90 NEXT L
100 C = RND(4) + 4
110 B = RND(4) + 4
120 PAINT (128,96), C,B
130 FOR K=1 TO 399: NEXT K
140 GOTO 50

```

Execute, admire e se extasie!!!

Para finalizar este capítulo tenho uma grande surpresa com o seguinte programa, algo assim como a sua casa, no sítio, uma garagem, o sol, etc.

Aliás neste programa aparecem quase todas as peculiares e importantes instruções pertencentes a "língua dos P" que estudamos até agora.

```

10 PMODE 1,1
20 PCLS
30 SCREEN 1,0
40 PCLS 3
50 COLOR 1,0
60 CIRCLE (200,28), 17
70 PAINT (200, 28), 2, 1
80 LINE (100,185) - (180,125), PSET, B
90 LINE - (140,90), PSET
100 LINE - (100,125), PSET
110 PAINT (135,115), 4, 1
120 LINE (110,160) - (125,130), PSET, B
130 LINE (155,160)-(170,130), PSET, B
140 PSET (134,157,1)
150 PAINT (120,180),0,1
160 LINE (130,130) - (149,185), PSET, B
170 LINE (101,135)-(41,185), PSET, B
180 LINE (91,140) - (51,185), PSET, B
190 PAINT (55,138), 0, 1
200 PAINT (89,183), 4, 1
210 FOR K=1 TO 399: NEXT K
220 PAINT (89,183),2,1
230 FOR K=1 TO 399: NEXT K
240 PAINT (89,155),4,1
250 GOTO 220

```

# ACERTANDO NA FIGURA

Neste capítulo vamos apresentar-lhe um jogo no qual testa-se a sua habilidade de lembrar o posicionamento de formas geométricas circulares ou elípticas.

Quatro contornos são desenhados próximos ao topo da tela e identificados pelos números 1,2,3 e 4 (veja a Fig. 47).

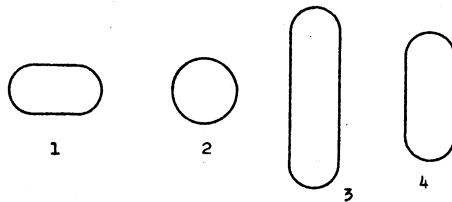


Figura 47

Uma quinta figura é desenhada bem em baixo da tela. A sua forma coincide com uma das quatro exibidas acima.

A finalidade do jogo é você responder qual é o número da figura de cima que é igual aquela apresentada em baixo.

Você deve teclar este número e só tem uma possibilidade para tentar acertar!!!

As quatro figuras mostradas no topo são apagadas antes que apareça a quinta figura.

Somente os números abaixo das figuras permanecem ativados e bem visíveis.





```

180 PSET(158,60): PSET(160,59): PSET(162,59)
190 LINE (164,60) - (164,65), PSET
200 PSET (162,62): PSET (158,65)
210 PSET (160,66): PSET (162,66)
215 FOR K=1 TO 599: NEXT K
220 LINE (206,60) - (206,64), PSET
230 LINE (208,64) - (210,64), PSET
240 LINE (212,60) - (212,66), PSET
245 FOR K=1 TO 599: NEXT K
250 REM AGORA VAMOS OBTER OS "CÍRCULOS" JUNTO COM UM ALERTA
    SONORO
260 PAINT (58,30),C,C: FOR T=1 TO 7: SOUND T,2: NEXT T
270 PAINT (108,30),C,C: FOR T=8 TO 15: SOUND T,3: NEXT T
280 PAINT (158,30),C,C: FOR T=16 TO 23: SOUND T,4: NEXT T
290 PAINT (208,30),C,C: FOR T=250 TO 255: SOUND T,3: NEXT T
300 REM UMA PAUSA PARA DEPOIS APAGAR TUDO
310 FOR K=1 TO 599 : NEXT C
320 REM VAMOS AGORA APAGAR AS FIGURAS
330 PAINT (58,30),5,5
340 PAINT (108,30),5,5
350 PAINT (158,30),5,5
360 PAINT (208,30),5,5
370 REM O TRS-2 VAI ESCOLHER UMA DAS QUATRO FORMAS
380 E(5) = .3 * RND(4) + .3 → note que não especificamos
390 CIRCLE(128,162),13,C,E(5)     a dimensão da variável in-
400 PAINT (128,162), C,C         dexada E(I) pois o índice
                                maior é menor do que 11
410 REM DESENHO DA DÚVIDA CRUEL
420 LINE (152,164) - (158,164),PSET
430 LINE (152,160) - (158,160), PSET
440 PSET(174,154): PSET(176,152): LINE(178,150) - (182,150),
    PSET
450 PSET(184,152): PSET(186,154): PSET(186,156)
460 PSET(184,158): LINE(182,160) - (182,164), PSET
470 PSET(182,170)

```

este é o 3

este é o 4

```

480 REM É AGORA QUE O TRS-2 VAI FAZER O TESTE SE VOCÊ
    ACERTOU O "CHUTE"
490 R$ = INKEY$: IF R$ = "" THEN 490
500 N = VAL(R$)
510 IF E(5) = E(N) THEN CLS: GOTO 550
520 CLS: PRINT "VOCÊ ERROU!!! TENDE DE NOVO."
530 FOR K=1 TO 699: NEXT K
540 GOTO 20
550 FOR K=1 TO 13
560 PRINT "PARABENS VOCÊ É UOOÓTIMO!!!"
570 FOR T=1 TO 10: SOUND T,1: NEXT T
580 NEXT K
590 PRINT: PRINT: PRINT
600 PRINT "APERTE QUALQUER TECLA E MOSTRE QUE NÃO FOI SORTE"
610 Q$ = INKEY$: IF Q$ = "" THEN 610
620 GOTO 20

```

Antes de executar aprenda o que é que se consegue com função STRING VAL que aparece na linha 500.

### 1ª Definição:

A função VAL converte uma STRING em um número.

Por exemplo se temos o programa

```

10 N$ = "909X"
20 N = VAL (N$)
30 PRINT N

```

ao executá-lo sai impresso 909.

Dê um RUN e confirme isto.

Vamos precisar adiante da função STRING STR\$ e aí vai a sua definição.

2ª Definição:

A função STR\$ (número) faz exatamente o contrário da VAL ("STRING") isto é transforma um número em uma STRING.

Teclae e execute o seguinte programinha para se convencer disto

```
1Ø N$ = STR$ (1313)
```

```
2Ø PRINT N$
```

A resposta exibida na tela é o número 1313?

Sem dúvida que é...

Bem, agora após estas informações você está autorizado(a) a executar o programão e ver se é bom de olho...



# A PODEROSA INSTRUÇÃO DRAW

Nos capítulos anteriores aprendemos a manipular cores, linhas, circunferências, etc.

A partir deste momento será apresentado a você dedicado(a) e estudioso(a) leitor(leitora) a mais poderosa instrução para a parte gráfica do seu TRS-2 e que permite de uma forma bem mais simples (assim acho...) elaborar desenhos na tela.

Refiro-me a instrução DRAW.

Com a instrução DRAW pode-se:

- 1) Especificar um ponto na tela como início para o desenho.
- 2) Especificar em que direção e de que tamanho deve ser o desenho.
- 3) Mudar a direção de retas consecutivas a serem desenhadas.
- 4) Voltar ao ponto inicial (origem) para desenhar uma nova reta.
- 5) Escolher a cor para o desenho.
- 6) Incluir variáveis alfanuméricas ou seja STRING como definições para comandos de movimento.
- 7) Incluir subSTRINGS para desenhos intermediários dentro da STRING principal do DRAW.
- 8) Concatenar STRINGS para o desenho.

Pois é, se com a instrução DRAW dá para fazer tudo isto, realmente ela deve ser a mais poderosa arma gráfica.

Estude tudo o que vem pela frente com cuidado redobrado pois as vantagens que obterá em função disto serão inúmeras!!!

A forma geral da instrução DRAW é:

DRAW "uma linha definindo uma STRING ou seja uma cadeia de caracteres"

É nesta cadeia de caracteres (STRING) que segue a instrução DRAW que estará toda a informação que possibilitará obter algumas das ações há pouco citadas.

Nesta linha pode-se ter:

#### I) Comandos de movimento

M → para mover para uma nova posição ("move")

U → para cima ("up")

D → para baixo ("down")

L → para a esquerda ("left")

R → para a direita ("right")

veja a Figura 48a

E → ângulo de  $45^\circ$

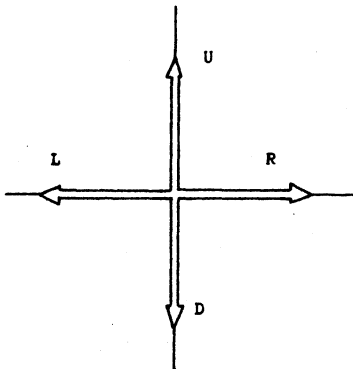
F → ângulo de  $135^\circ$

G → ângulo de  $225^\circ$

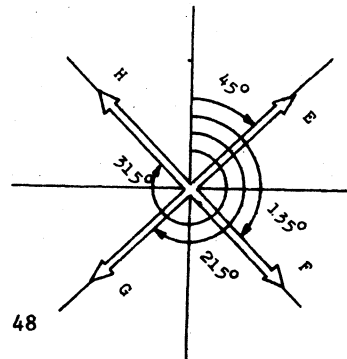
H → ângulo de  $315^\circ$

veja a Figura 48b.

X → executa uma subSTRING e volta a STRING principal



a)



b)

Figura 48

## II) Modos

C → cor ("color")

A → ângulo ("angle")

S → escala ("scale")

## III) Opções

N → Sem precisar atualizar a posição do desenho (veja o Capítulo 19)

B → Espaço em branco (não desenha apenas desloca)

O.I. - O.I. - O.I. - O.I.

Se uma linha for uma STRING constante deve obrigatoriamente aparecer entre aspas.

É necessário colocar sempre a opção B antes do comando de movimento M para que não surjam linhas não requeridas.

Inicialmente vamos mostrar como é possível obter a seguinte representação com o uso da instrução DRAW.

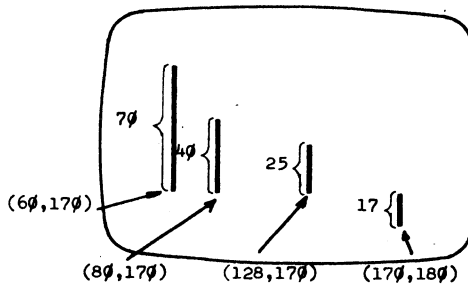


Figura 49

1Ø PMODE 0,1

2Ø PCLS

3Ø SCREEN 1,0

40 REM É AGORA QUE SERÁ ENSINADO AO TRS-2 COMO DESENHAR AS QUATRO RETAS

50 DRAW "BM 60,170; U 70"

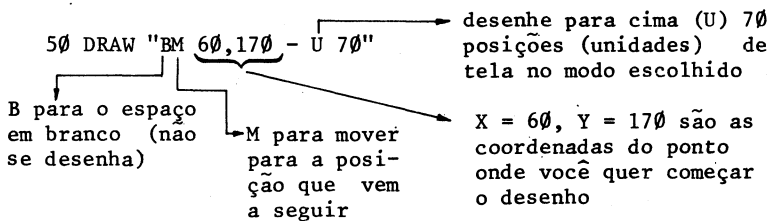
60 DRAW "BM 80,170; U 40"

70 DRAW "BM 128,170; U 25"

80 DRAW "BM 200,190; U 17"

90 GOTO 90

Vamos explicar com detalhes a linha 50 com o que também se tornará claro o que se quer com as linhas 60, 70 e 80.



Execute e constate o que está indicado na Figura 49.



OIZÃO

Tire o B antes do M nas linhas 60, 70 e 80 e veja que se você executar o programa terá como saída o que se mostra na Fig. 50.

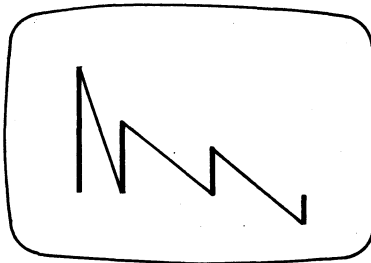


Figura 50

e como se vê ninguém pediu as linhas retas inclinadas.



Qual é a razão disto?

Resp.: Quando você escreve a linha 60 com  
DRAW "M 80,170; U 40"

o B não está presente antes do M e neste movimento do último ponto da reta desenhada com a instrução da linha 50(60,170) para o novo ponto (80,170) surge uma linha reta que não foi perdida.

O mesmo ocorre entre a 2ª e 3ª linhas retas e entre a 3ª e a 4ª linhas retas.

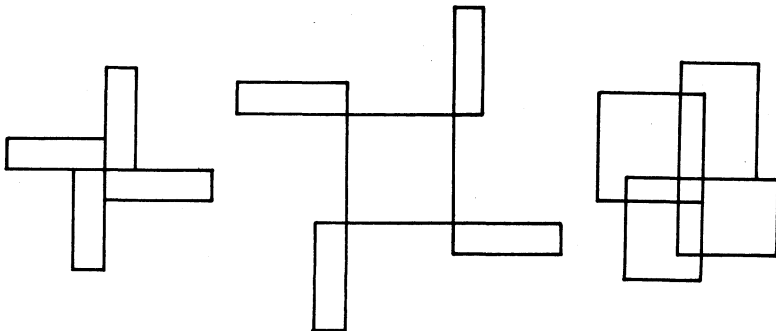
Como deu para perceber a omissão do B pode ser totalmente indesejada pois trará um traço contínuo.

A sua criatividade está despertada e você está todo(a) agitado(a) para usar todas as opções de movimento.

É por isto que vamos apresentar um programa que permite desenhar as lindas "espirolaterais" onde vão aparecer mais algumas opções.

Na Figura 51a) temos uma "espirolateral" de 3ª ordem que é de certa forma similar a uma espiral porém com a diferença que ela é obtida com o auxílio de linhas retas e não com uma curva "contínua, uniforme e bem "polida".

As espirolaterais podem ser de três tipos [veja as Figuras 51b) e c)] e são chamadas de 3ª ordem pois são obtidas com o auxílio de 3 movimentos.



a) Os retângulos são tangentes

Figura 51  
b) Existe um "buraco" no centro

c) Há superposição

Suponhamos que o ponto inicial escolhido seja  $(120,60)$  e que as distâncias percorridas nos três movimentos sejam respectivamente iguais a 40, 60 e 100, o que se obtém com uma instrução do tipo

`DRAW "BM 120,60; R 40; D 60; L 100" ?`

Resp.: Aquilo que está mostrado na Figura 52

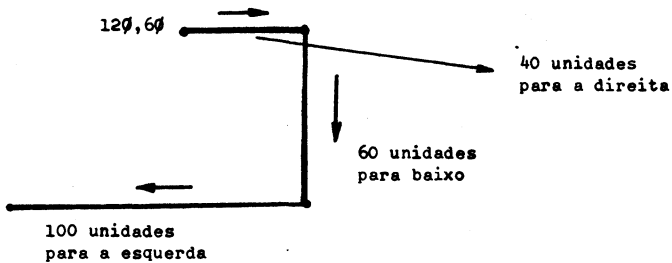


Figura 52

Bem aí vai o programa usando concatenação adoidadamente.

5 CLS

10 PRINT "VOCÊ DEVE ENTRAR COM AS TRÊS DISTÂNCIAS"

20 FOR Z=1 TO 799: NEXT Z

30 CLS

40 INPUT "A PRIMEIRA DISTÂNCIA:"; D1\$

50 INPUT "A SEGUNDA DISTÂNCIA:"; D2\$

60 INPUT "A TERCEIRA DISTÂNCIA:"; D3\$

70 REM VAMOS AJUSTAR A TELA PARA O MODO GRÁFICO

80 PMODE 3,1

90 PCLS

100 SCREEN 1,1

110 REM VAMOS AGORA CONCATENAR E ARMAR AS NOSSAS INSTRUÇÕES DE MOVIMENTO

120 A\$ = "R"+D1\$ + "D"+D2\$ + "L"+D3\$ → 1º deslocamento com 3 movimentos

130 B\$ = "U" + D1\$ + "R" + D2\$ + "D" + D3\$ → 2º deslocamento com 3 movimentos

```

140 C$ = "L"+D1$ + "U"+D2$ + "R"+D3$ → 39 deslocamento com
                                     3 movimentos
150 D$ = "D"+D1$ + "L"+D2$ + "U"+D3$ → 49 deslocamento com
                                     3 movimentos
160 REM É AGORA O GRANDE MOMENTO DE DESENHAR A ESPIROLATE-
    RAL
170 DRAW "BM 100,60" + A$: FOR T=1 TO 5: SOUND T,2: NEXT T
180 FOR K=1 TO 599: NEXT K
190 DRAW B$: FOR T=1 TO 5: SOUND T,3: NEXT T
200 FOR K=1 TO 599: NEXT K
210 DRAW C$: FOR T=1 TO 5: SOUND T,4: NEXT T
220 FOR K=1 TO 599: NEXT K
230 DRAW D$: FOR T=1 TO 20: SOUND T,1: NEXT T
240 R$ = INKEY$: IF R$ = "" THEN 240
250 IF LEFT$(R$,1) <> "F" THEN 5
260 END

```

→ Veja no Capítulo (7) o significado da função `STRING LEFT$`, mas por enquanto tecle F para parar ou qualquer outro caractere para continuar.

No programa fica por sua conta destrinchar alguma instrução que não entendeu entretanto como concatenação foi algo muito usado (aliás já fizemos isto no Capítulo 5 e ficamos na "moita"...) vamos a mais detalhes.

Quando se diz que o TRS-2 faz gráficos, faz contas intermináveis cálculos cheio de complexidade, emite sons você aceita, não é?

Porém se alguém lhe disser que o seu TRS-2 sabe manipular textos ou seja trabalhos com STRINGS você fica meio na dúvida, correto?

Pois tire logo esta sua última dúvida aprendendo que o TRS-2 pode emendar ou "somar" STRINGS, comparar STRINGS (o que é importante para se ter uma ordem alfabética), etc.

Chama-se então de *concatenação* a operação de juntar ou se você assim quiser somar STRINGS.

Não pense que é possível diminuir, multiplicar ou dividir uma STRING por outra.

Isto também já seria demais!!!

Tecele o seguinte "programinha"

```

10 X$ = "COMPREI O"
20 Y$ = "□ ESPETACULAR"
30 Z$ = "□ "
40 W$ = " TRS-2
50 A$ = X$ + Y$ + Z$ + W$ → aĩ está a concatenação
60 PRINT A$

```

Execute o "programinha" e veja que sai escrito na tela

```

COMPREI O ESPETACULAR TRS-2

```

Nas linhas 120, 130, 140 e 150 temos a operação de concatenação assim como na linha 170 onde concatenou-se "BM 100,60" com A\$.

Execute agora o programa "ESPIROLATERAL" entrando no começo com 40, 60 e 100 e obtendo um desenho semelhante ao da Figura 51a.

Mude depois estas entradas e veja se obtêm as duas outras representações.

# EXPLORANDO NOVOS CAMINHOS

Além de podermos desenhar para cima, para a esquerda, para a direita e para baixo com a instrução DRAW pode-se desenhar com movimentos sob o ângulo de  $45^\circ$ ,  $135^\circ$ ,  $225^\circ$  e  $315^\circ$  (veja a definição da instrução DRAW que foi dada no Capítulo 18).

Para começar vamos fazer um pequeno programa que permita exibir o que está mostrado na Figura 53.

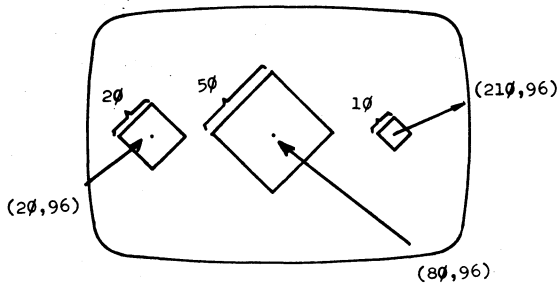


Figura 53

1φ PMODE 0,1

2φ PCLS

3φ SCREEN 1,0

4φ DRAW "BM 2φ,96; E2φ; F2φ; G2φ; H2φ"

5φ DRAW "BM 8φ,96; E5φ; F5φ; G5φ; H5φ"

6φ DRAW "BM 21φ,96; E1φ; F1φ; G1φ; H1φ"

7φ GOTO 7φ

O movimento é inicialmente sob ângulo de  $45^\circ$  depois de  $135^\circ$ , a seguir de  $225^\circ$  e finalmente sob o ângulo de  $315^\circ$

Execute e constate que sai o que está mostrado na Figura 53.

Os comprimentos das linhas "retas" desenhadas pelos comandos de ângulo podem surpreendê-lo(1a) um pouco pois o valor especificado para comprimento de uma diagonal é igual ou seja é o mesmo que o da sua projeção (veja as Figuras 54 a, b) c) e d).

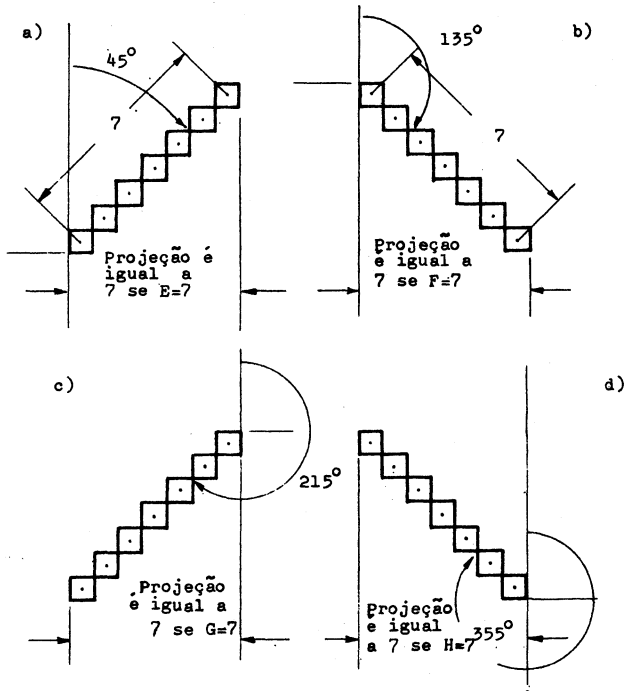


Figura 54



O.I. Cuidado então pois no seu TRS-2 a projeção é igual ao comprimento da linha reta que se manda traçar com os comandos de ângulo.

Para destacar este comportamento um "tanto estranho" do comprimento de linhas, vamos apresentar um programa que permite obter como saída os desenhos indicados na Figura 55.

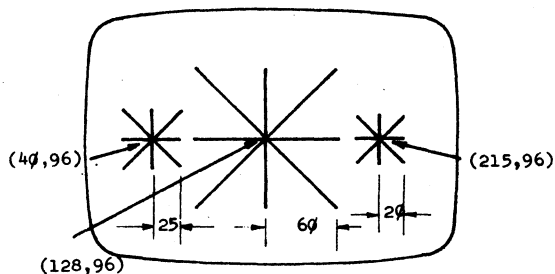


Figura 55

Aí vai o programa que podemos chamar de "FEIXE DE RETAS"

```

5 REM PROGRAMA FEIXE DE RETAS
10 PMODE 0,1
20 PCLS
30 SCREEN 1,0
40 DRAW "BM 128,96; U60"
50 DRAW "BM 128,96; R60"
60 DRAW "BM 128,96; D60"
70 DRAW "BM 128,96; L60"
80 DRAW "BM 128,96; E60"
90 DRAW "BM 128,96; F60"
100 DRAW "BM 128,96; G60"
110 DRAW "BM 128,96; H60"
115 FOR K=1 TO 499: NEXT K
120 DRAW "BM 40,96; U25"
130 DRAW "BM 40,96; R25"
140 DRAW "BM 40,96; D25"
150 DRAW "BM 40,96; L25"
160 DRAW "BM 40,96; E25"
170 DRAW "BM 40,96; F25"
180 DRAW "BM 40,96; G25"
190 DRAW "BM 40,96; H25"
200 FOR K=1 TO 499: NEXT K

```

aqui sai o feixe maior

aqui sai o feixe pequeno e da esquerda

```

210 DRAW "BM 215,96; U20"
220 DRAW "BM 215,96; R20"
230 DRAW "BM 215,96; L20"
240 DRAW "BM 215,96; D20"
250 DRAW "BM 215,96; E20"
260 DRAW "BM 215,96; F20"
270 DRAW "BM 215,96; G20"
280 DRAW "BM 215,96; H20"
290 GOTO 290

```

aqui sai o menor dos feixes,  
o da direita

Dê um RUN e veja se sai algo semelhante ao que se mostra na Figura.

Na realidade este último programa ficou muito grande para sair tão pouca coisa na tela e o seu TRS-2 tem um recurso muito importante dentro da instrução DRAW (aliás já citado no Capítulo 18) com o que se diminuirá em muito o trabalho de teclar.

A instrução DRAW admite a opção N com a qual automaticamente volta-se a posição inicial usada no último comando DRAW.

Desta forma usando este N as linhas 40 a 110, 120 a 190 e 210 a 280 podem ser transformadas em uma única e o nosso programa "FEIXE DE RETAS" toma o seguinte aspecto.

```

10 PMODE 0,1
20 PCLS
30 SCREEN 1,0
40 DRAW "BM 128,96; NU60; NR60; ND60; NL60; NE60; NF60; NG60;
NH60"
50 FOR K=1 TO 499: NEXT K
60 DRAW "BM 40,96; NU25; NR25; ND25; NL25; NE25; NF25; NG25;
NH25"
70 FOR K=1 TO 499: NEXT K
80 DRAW "BM 215,96; NU20; NR20; ND20; NL20; NE20; NF20; NG20;
NH 20"
90 GOTO 90

```



Execute, veja que dá o mesmo resultado anterior (Figura 55) e atribua a partir deste ponto o devido valor a esta letra N providencialmente colocada antes dos diversos movimentos.

Uma outra coisa que você pode inserir dentro da sua instrução DRAW é a cor através da forma geral

DRAW "CX; ....."

aquí X é o código da cor ou seja um valor inteiro de 0 a 8

Uma última mudança no programa "FEIXE DE RETAS" poderia ser aquela de tornar as linhas inclinadas (diagonais) nos feixes menores de cor ciano e no feixe maior de cor lilás.

É o que se consegue com o programa que vem a seguir

```

10 PMODE 3,1
20 PCLS
30 SCREEN 1,1
40 DRAW "C8; BM 128,96; NU55; NR55; ND55; NL55"
45 DRAW "C7; BM 128,96; NE55; NF55; NG55; NH55" cor lilás
50 FOR K=1 TO 499 : NEXT K
60 DRAW "C8; BM 35,96; NU25; NR25; ND25; NL25"
65 DRAW "C6; BM 35,96; NE25; NF25; NG25; NH25"
70 FOR K=1 TO 499: NEXT K
80 DRAW "C8; BM 215,96; NU20; NR20; ND20; NL20" cor ciano
85 DRAW "C6; BM 215,96; NE20; NF20; NG20; NH20"
90 GOTO 90

```

Cada N diz ao micro para voltar ao ponto original para traçar a próxima reta

Execute e comprove com os próprios olhos o lindo colorido que foi produzido pela oportuna introdução de C7 (linha 45) e C6 (linhas 65 e 85).

Note que para não haver mistura de cores usamos a cor 8 (laranja) nas outras retas.

Pequenos probleminhas para você exercitar-se no seu TRS-2 atencioso(a) leitor(leitora):

1) Use a instrução DRAW para desenhar um quadrado com o canto inferior da esquerda no ponto  $(50,130)$  e com o lado tendo 80 unidades de comprimento.

Além disto desenhe também a diagonal que sai do canto inferior da esquerda.

2) Escreva um programa usando as instruções DRAW e CIRCLE entre outras para obter na tela o desenho mostrado na Figura 56 se possível com as circunferências tendo cor diferente dos quadrados e dos losangos.

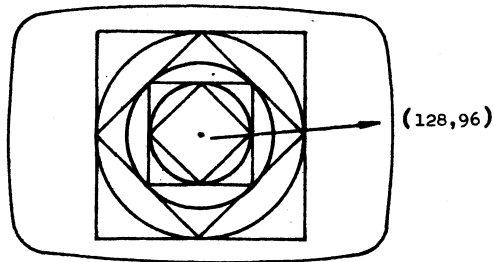


Figura 56

# SUBCADEIA DENTRO DE UMA LINHA DRAW

Você já sabe como trabalha uma subrotina em um programa para o seu TRS-2

É possível executar algo semelhante dentro da cadeia (STRING) que vem logo após a instrução DRAW.

O comando de movimento X permite que você execute uma subcadeia (subSTRING) dentro de uma cadeia (STRING).

Para mostrar como isto funciona vamos elaborar um programa que permite apresentar na tela a saída mostrada na Figura 57.

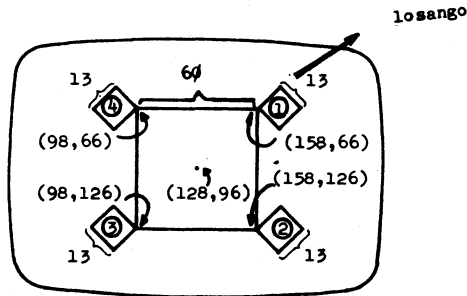


Figura 57

Aí vai o programa:

```
10 PMODE 4,1
20 PCLS
30 SCREEN 1,0
40 D$ = "E13; F13; G13; H13"
```

45 L\$ = "H13; G13; F13; E13"

50 DRAW "BM 98,66; R60; XD\$; D60; XD\$; LG0; XL\$; U60; XL\$;"

60 GOTO 60

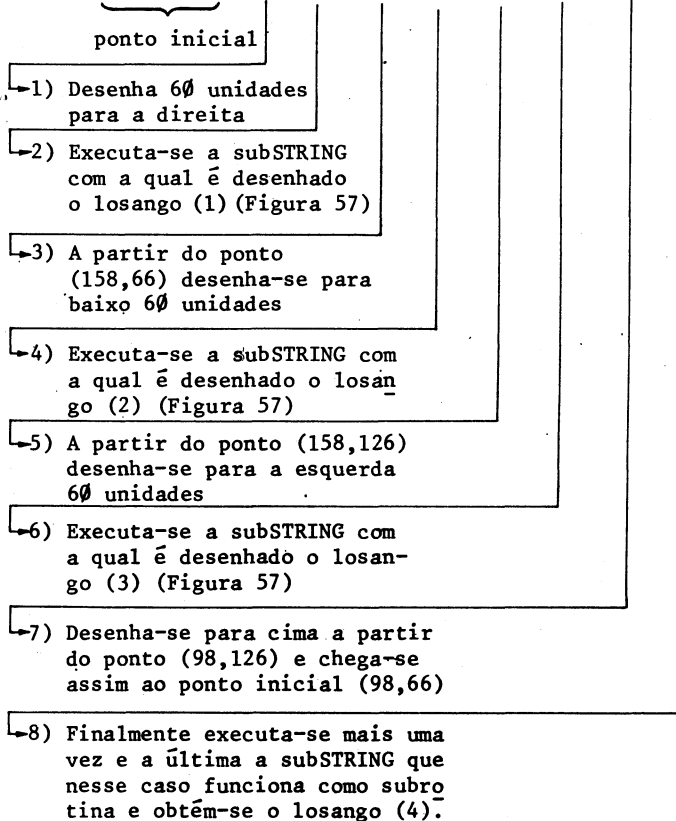
Execute e veja se sai o que está na Figura 57.

Saiu?

Formidável este X!!!

Aí estão os detalhes do que se obtém com a linha 50.

50 DRAW "BM 98,66; R60; XD\$; D60; XD\$; L60; XL\$; U60; XL\$;"





O.I. - O.I. - O.I. - O.I. - O.I.

Quando você utiliza a instrução DRAW não é necessário sempre separar os comandos de movimento por ponto e vírgula (;).

Temos até agora feito esta separação com finalidade didática para tornar mais óbvio e legível cada comando de movimento.

Portanto você pode omitir os pontos e vírgula como separadores para a maioria dos comandos.

O que nunca pode ser feito é a omissão do ponto e vírgula (;) após o comando de subSTRING X.

É aceita então a instrução

```
50 DRAW "BM 98,66 R60 XD$; D60XD$; L60XL$; U60XL$;"
```

Substitua a mesma no programa e teste para ter a certeza que o seu efeito é o mesmo que tínhamos anteriormente com a diferença que não precisamos teclar tanto.

Vamos agora mostrar uma outra forma de efetuar a concatenação dentro da instrução DRAW, com o que fica mais fácil a leitura do programa e as eventuais modificações que devem ser feitas no mesmo.

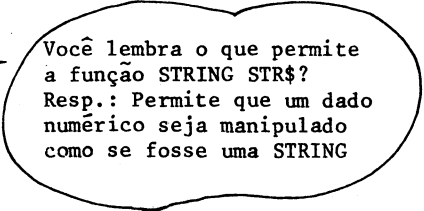
Como exemplo disto vamos desenhar um retângulo e introduzir em um laço FOR-NEXT a possibilidade de mudar a cor do mesmo de amarelo para azul e depois para o vermelho.

Após isto vamos apagar este retângulo usando a cor do fundo.

Aí vai o programa:

```
10 PMODE 3,1
20 PCLS
30 SCREEN 1,0
40 A$ = "BM 70,110 U50 R80 D50 L80"
50 B$ = "C1 BM 70,110 U50 R80 D50 L80"
60 REM AQUI SE ATRIBUI OS CÓDIGOS DAS CORES COMO STRINGS
```

```
70 FOR J=2 TO 4
80 C$(J) = "C" + STR$(J)
90 NEXT J
100 FOR K=2 TO 4
110 DRAW C$(K) + A$
120 FOR Z=1 TO 27: NEXT Z
130 DRAW B$
140 FOR Z=1 TO 27: NEXT Z
150 NEXT K
160 GOTO 100
```



Você lembra o que permite a função STRING STR\$?  
Resp.: Permite que um dado numérico seja manipulado como se fosse uma STRING

Estã gostando do pisca-pisca retangular?

Uootimo!!!

# MOVIMENTO RELATIVO

Nós já usamos bastante o comando M para mover um específico ponto na tela com o intuito de desenhar uma figura geométrica.

Pode-se entretanto também utilizar o comando M para mover a uma distância específica *relativamente* a última posição utilizada ou seja do último desenho.

A forma geral do comando M é

$$M \quad \underline{+} \quad H, \quad \underline{+} \quad V$$

onde H é um número especificando a distância para se deslocar da posição atual X.

Se o deslocamento H é antecedido por um sinal "+" (mais) a posição X é incrementada de H.

Se o deslocamento H é antecedido por um sinal "-" (menos) a posição X é decrementada (diminuída) da quantidade especificada.

A mesma explicação se aplica ao deslocamento vertical V em relação a posição atual (última) Y.

Para deslocamentos positivos o sinal pode ser omitido apenas para o V.

Como exemplo vamos elaborar um programa que permita obter a representação mostrada na Figura 58.

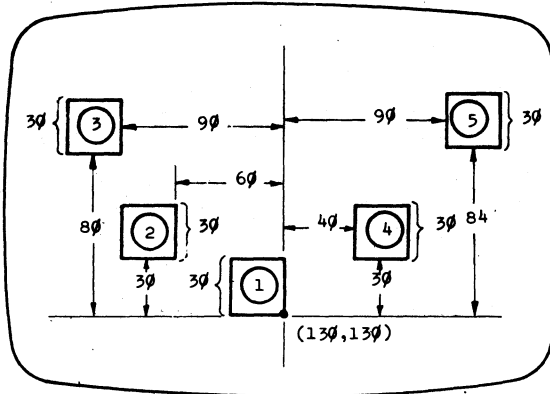


Figura 58

Aí vai o programa:

```

10 PMODE 3,1
20 PCLS
30 SCREEN 1,0
40 REM VAMOS DESENHAR O QUADRADO ORIGINAL E OS OUTROS
50 DRAW "BM 130,130 U30 L30 D30 R30" → quadrado (1)
60 FOR K=1 TO 399: NEXT K
70 DRAW "BM-60,-30 U30 L30 D30 R30" → quadrado (2)
75 FOR K=1 TO 399: NEXT K
80 DRAW "BM -30, -50 U30 L30 D30 R30" → quadrado (3)
85 FOR K=1 TO 399: NEXT K
90 DRAW "BM + 160, 50 U30 L30 D30 R 30" → quadrado (4)
100 FOR K=1 TO 399: NEXT K
110 DRAW "BM+50, -54 U30 L30 D30 R30" → quadrado (5)
115 FOR K=1 TO 399: NEXT K
120 GOTO 20

```

Execute e constate que sai na tela o que está mostrando na Figura 58.

Uma das mais importantes aplicações da instrução gráfica DRAW é a de poder criar um texto que possa ser apresentado e lido por você em modo gráfico.



Vamos agora imprimir uma linda circunferência e abaixo dela vamos escrever "CIRCUNFERÊNCIA" utilizando para isto letras obtidas com o auxílio da instrução DRAW.

Cada letra será construída em um reticulado 12x12 (veja a Figura 59) e preencherá uma matriz 8x8.

Para escrever todas as letras inicialmente se apresenta uma subrotina que permite obter todas as letras.

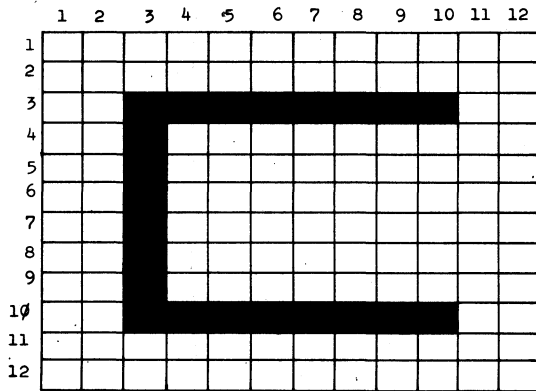


Figura 59

```

1000 REM SUBROTINA PARA DESENHAR LETRAS
1010 L$(1) = "U8R8D4L8BR8D4BL8" → A
1020 L$(2) = "U8R6F2D2L8BR8D2G2L6" → B
1030 L$(3) = "U8R8BD8L8" → C
1040 L$(4) = "U8R6F2D4G2L6" → D
1050 L$(5) = "U8R8BD4L8BR8BD4L8" → E
1060 L$(6) = "U8R8BD4BL4L4BD4" → F
1070 L$(7) = "U8R8BD4L4BR4D4L8" → G
1080 L$(8) = "U8BR8D8BU4L8BD4" → H
1090 L$(9) = "BU8R8BL4D8BR4L8" → I
1100 L$(10) = "U4BU4BR8D8L8" → J
1110 L$(11) = "U8BR8G4L4BR4F4BL8" → K
1120 L$(12) = "U8BR8BD8L8" → L

```

1130 L\$(13) = "U8F4E4D8BL8" → M  
 1140 L\$(14) = "U8F8U8BG8" → N  
 1150 L\$(15) = "U8R8D8L8" → O  
 1160 L\$(16) = "U8R8D4L8BD4" → P  
 1170 L\$(17) = "U8R8D8H4BF4L8" → Q  
 1180 L\$(18) = "U8R8D4L8BR4F4BL8" → R  
 1190 L\$(19) = "BU4U4R8BD4L8BR8D4L8" → S  
 1200 L\$(20) = "BU8R8BL4D8BL4" → T  
 1210 L\$(21) = "U8BR8D8L8" → U  
 1220 L\$(22) = "BU8D4F4E4U4BG8" → V  
 1230 L\$(23) = "U8BR8D8H4G4" → W  
 1240 L\$(24) = "BU8F8BU8G8" → X  
 1250 L\$(25) = "BU8F4E4BG4D4BL4" → Y  
 1260 L\$(26) = "BU8R8G8R8BL8" → Z  
 1270 RETURN

Vamos então ao programa:

10 CLEAR 500  
 20 DIM L\$(27) → especifica-se aqui a dimensão da  
 matriz STRING ou seja matriz alfa  
 30 PMODE 4,1 numérica  
 40 PCLS  
 50 CIRCLE (128,70),60  
 60 SCREEN 1,0  
 70 GOSUB 1000  
 80 DRAW "BM 44,160" + L\$(3) → C  
 90 DRAW "BM + 12,0" + L\$(9) → I  
 100 DRAW "BM +12,0" + L\$(18) → R  
 110 DRAW "BM + 12,0" + L\$(3) → C  
 120 DRAW "BM + 12,0" + L\$(21) → U  
 130 DRAW "BM + 12,0" + L\$(14) → N  
 140 DRAW "BM + 12,0" + L\$(6) → F  
 150 DRAW "BM + 12,0" + L\$(5) → E  
 160 DRAW "BM + 12,0" + L\$(18) → R

```

17Ø DRAW "BM + 12,Ø" + L$(5) ———→ E
18Ø DRAW "BM + 12,Ø" + L$(14) ———→ N
19Ø DRAW "BM + 12,Ø" + L$(3) ———→ C
20Ø DRAW "BM + 12,Ø" + L$(9) ———→ I
21Ø DRAW "BM + 12,Ø" + L$(1) ———→ A
22Ø GOTO 22Ø

```



O.I. Aparece na linha 1Ø a instrução CLEAR.

A forma geral da instrução CLEAR é

CLEAR n, h

e com ela reserva-se n bytes do espaço para armazenamento de variáveis STRING.

Com o h especifica-se, se for necessário o mais alto endereço em BASIC.

Com a instrução de número de linha 1Ø reservou-se um espaço para 500 caracteres.

Podemos simplificar um pouco este programa escrevendo as linhas 8Ø até 21Ø em uma só ou seja:

```

8Ø DRAW "BM 44,16Ø" + L$(3) + "BR12" + L$(9) + "BR12" + L$(18)
+ "BR12" + L$(3) + "BR12" + L$(21) + "BR12" + L$(14)
+ "BR12" + L$(6) + "BR12" + L$(5) + "BR12" + L$(18) +
"BR12" + L$(5) + "BR12" + L$(14) + "BR12" + L$(3) +
"BR12" + L$(9) + "BR12" + L$(1)

```

Porém desta última condensação você já percebeu que o movimento para a direita pode ser incluído dentro de cada caractere.

Assim por exemplo podemos escrever o A como

L\$(1) = "U3R8D4L3BR8D4BR4" —→ veja a Figura 60a

Já para o B temos:

L\$(2) = "U8R6F2D2L8BR8D2G2L6BR12" —→ veja a Figura 60b

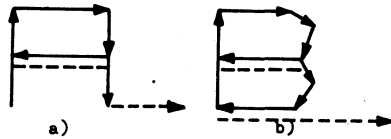


Figura 60

Agora a subrotina para a obtenção dos caracteres de texto através da instrução DRAW é:

```

1000 REM SUBROTINA REVISADA PARA DESENHAR LETRAS
1010 L$(1) = "U8R8D4L8BR8D4BR4" → A
1020 L$(2) = "U8R6F2D2L8BR8D2G2L6BR12" → B
1030 L$(3) = "U8R8BD8L8BR12" → C
1040 L$(4) = "U8R6F2D4G2L6BR12" → D
1050 L$(5) = "U8R8BD4L8BD4R8BR4" → E
1060 L$(6) = "U8R8BD4L8BD4BR12" → F
1070 L$(7) = "U8R8BD4L4BR4D4L8BR12" → G
1080 L$(8) = "U8BR8D8BU4L8BD4BR12" → H
1090 L$(9) = "BU8R8BL4D8BL4R8BR4" → I
1100 L$(10) = "U4BU4BR8D8L8BR12" → J
1110 L$(11) = "U8BR8G4L4BR4F4BR4" → K
1120 L$(12) = "U8BD8R8BR4" → L
1130 L$(13) = "U8F4E4D8BR4" → M
1140 L$(14) = "U8F8U8BD8BR4" → N
1150 L$(15) = "U8R8D8L8BR12" → O
1160 L$(16) = "U8R8D4L8BD4BR12" → P
1170 L$(17) = "U8R8D8H4BG4R8BR4" → Q
1180 L$(18) = "U8R8D4L8BR4F4BR4" → R
1190 L$(19) = "BU4U4R8BD4L8BR8D4L8BR12" → S
1200 L$(20) = "BU8R8BL4D8BR8" → T
1210 L$(21) = "U8BR8D8L8BR12" → U
1220 L$(22) = "BU8D4F4E4U4BD8BR4" → V
1230 L$(23) = "U8BR8D8H4G4BR12" → W
1240 L$(24) = "E8BLSF8BR4" → X

```

```

1250 L$(25) = "BU8F4E4BC4D4BR8" —————> Y
1260 L$(26) = "BU8R8G8R8BR4" —————> Z
1270 L$(27) = "BR12" —————> para se ter um espaço em branco
1280 RETURN

```

A subrotina revisada torna muito mais simples o programa.

```

10 CLEAR 500
20 DIM L$(27)
30 PMODE 4,1
40 PCLS
50 CIRCLE (128,70),60
60 SCREEN 1,0
70 GOSUB 1000
80 DRAW "BM 44,160" + L$(3) + L$(9) + L$(18) + L$(3) +
    L$(21) + L$(14) + L$(14) + L$(5) + L$(18) + L$(5) +
    L$(14) + L$(3) + L$(9) + L$(1)
90 GOTO 90

```

Execute e verifique com satisfação um texto no modo gráfico.

Isto não é nada fácil de conseguir nos outros micros, porém como você constata é bem simples no seu TRS-2

Para mostrar a você mesmo que entendeu tudo e é capaz de fazer coisas por si mesmo elabore um programa que permita obter a saída exibida na Figura 61, usando as instruções DRAW, CIRCLE, PSET, etc.

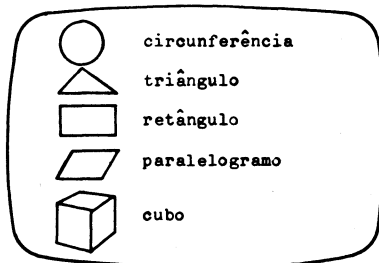


Figura 61



# MEMORIZANDO

No Capítulo 21 usamos uma subrotina para desenhar letras e agora vamos aproveitá-la para permitir que você participe de um jogo de rapidez ou melhor ação combinada da sua "cuca" e dos seus dedos.

Se quiser pode chamar o jogo de "*rápido de dedo*".

Você sabe que estamos usando um alfabeto de 26 letras e usamos 27 variáveis subscritas ou indexadas de L\$(1) a L\$(27) na nossa subrotina revisada.

A variável L\$(27) foi usada no Capítulo 21 para armazenar o espaço em branco e neste jogo será ignorado.

O programa irá escolher aleatoriamente 13 vezes, letras do alfabeto, e caberá a você responder qual é o número de cada letra (A é 1, B é 2, C é 3, ..., Z é 26).

Lembre que com L\$(1) desenha-se um A, com L\$(2) desenha-se um B, etc.

No programa temos um cronômetro que determina quanto tempo você leva para responder.

Naturalmente incluímos uma penalidade para você se for "molé" e ela depende também de quanto difere o número que você introduziu com a correta posição da letra no alfabeto.

Não esqueça que a subrotina a qual nos referimos é a revisada do Capítulo 21.

Divirta-se aprendendo a posição "numérica" das letras no alfabeto com o seguinte programa.

```

10 REM O JOGO DO ALFABETO
20 CLEAR 5000
30 DIM L$(27), P(13) → matriz unidimensional para o número de pontos que você obtém em cada resposta
40 CLS
50 PMODE 4,1
60 PCLS
70 SCREEN 1,0
80 GOSUB 10000
90 P = 0 → variável na qual será armazenada a sua contagem final
100 FOR J=1 TO 13
110 R = RND(26) → escolha aleatória da letra
120 DRAW "BM 120,90" + L$(R) → desenha-se a primeira letra
130 TIMER = 0 → cronômetro ajustado em zero
140 DRAW "BM 4,15; U8" + L$(27) → pronto para o 1º dígito
150 R$(1) = INKEY$: IF R$(1) = "" THEN 150
160 T1 = TIMER: TIMER = 0
170 DRAW "BM 4,15; U8BR12D8" → pronto para o 2º dígito
180 R$(2) = INKEY$: IF R$(2) = "" THEN 180
190 T2 = TIMER
200 N = 10*VAL(R$(1)) + VAL(R$(2)) → vá ao Capítulo 7 para lembrar o que faz a função STRING VAL. Entretanto é aí que é calculada a sua resposta
210 E = ABS(R-N)
220 T = INT((T1+T2)/6)
230 P(J) = T + 30 * E → Resposta em décimos de segundos
240 PCLS
250 NEXT J
260 CLS
265 FOR I=1 TO 32: PRINT "*"; : NEXT I
270 PRINT "A CONTAGEM EM CADA ETAPA FOI:"
280 FOR J=1 TO 13
290 P = P+P(J)
300 PRINT TAB(10); " "; J; " "; P(J)
310 NEXT J
320 FOR K=1 TO 2000: NEXT K: CLS

```

→ aqui se soma o tempo e a penalidade devida ao erro



```

33Ø PRINT "A CONTAGEM FINAL FOI:"
34Ø PRINT TAB(12); "P="; P
35Ø PRINT: PRINT
36Ø PRINT "APERTE QUALQUER TECLA PARA JOGAR NOVAMENTE"
37Ø R$ = INKEY$: IF R$ = "" THEN 37Ø
38Ø GOTO 6Ø~

```

Para uma participação correta no jogo leia um pouquinho mais.

### *I) Explicações sobre novas instruções*

1) O TRS-2 possui um cronômetro ("temporizador") que conta de Ø até 65535 quando então ele recomeça tudo de novo.

O contador para durante a operação do gravador ou da impressora.

A função TIMER usa um relógio para medir o tempo em unidades aproximadamente iguais a 1/6Ø do segundo.

Caso você queira saber o tempo que transcorreu em segundos basta dividir por 60, aliás você dispõe de  $\frac{65535}{60} \approx 1092$  segundos em cada ciclo do TIMER.

Tecele e execute o seguinte programinha para entender bem como funciona o TIMER.

Ah!!! Compare o resultado com o que marca o seu relógio principalmente se tiver um cronômetro.

```

1Ø PRINT "PENSE EM ALGO GOSTOSO"
2Ø PRINT: PRINT: PRINT
3Ø PRINT "APERTE QUALQUER TECLA QUANDO TERMINAR DE PENSAR"
4Ø R$ = INKEY$: IF R$ <> "" THEN 5Ø ELSE 4Ø
5Ø PRINT "VOCÊ PENSOU DURANTE"; INT(TIMER/6Ø); " □SEGUNDOS"
6Ø PRINT: PRINT: PRINT "ENTRE COM (S) SE QUISER CONTINUAR"
7Ø INPUT Q$: IF Q$ = "S" THEN 9Ø
8Ø END
9Ø TIMER = Ø
1ØØ GOTO 1Ø

```

Bem com este programinha ficam claras as instruções das linhas 130, 160 e 190.

2) Na linha 210 temos a função numérica ABS que toma o valor absoluto do argumento.

No nosso caso particular calcula-se o erro em valor absoluto entre a sua resposta (N) e a real posição da letra (R).

3) A função TAB(n) é usada para especificar a coluna na qual a impressão deve começar.

No TRS-2 a primeira coluna é numerada com zero, assim se você executar o mini programa

```
10 A$ = "TRS-2 - O MICRO"
```

```
20 PRINT TAB(13); A$
```

devido a instrução TAB na linha 20 a mensagem "TRS-2 - O MICRO" começa a ser impressa a partir da coluna numerada com 13 ou seja a décima quarta coluna.

Com esta explicação, não há mais o que duvidar sobre o que ocorre nas linhas 300 e 340.

## II) Explicações sobre o jogo

Lembre que velocidade e acurácia (precisão) são ambas muito importantes neste jogo.

Assim se aparecer a letra:

$$\left\{ \begin{array}{l} \text{A entre com } 01; \\ \text{B entre com } 02; \\ \text{C entre com } 03; \\ \vdots \\ \text{Z entre com } 26. \end{array} \right.$$

O aviso de que você deve entrar com a sua resposta é dado no canto superior da esquerda da sua tela.

Assim por exemplo se o TRS-2 escolher a letra N ou seja N = L\$(14) você já sabe que a letra N é a décima quarta e a entrada correta é 14 (veja as Figuras 62 e 63).

Entre inicialmente com o 1 e em seguida com o 4.

Não precisa apertar o ENTER pois está se usando o INKEY\$ para uma entrada, a mais rápida possível.

O "pronto" para entrar com o 1º dígito é indicado pelo que aparece neste canto

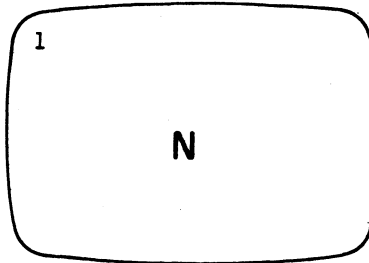


Figura 62

O "pronto" para entrar com o 2º dígito é indicado pelo segundo "1" que aparece no canto superior da esquerda

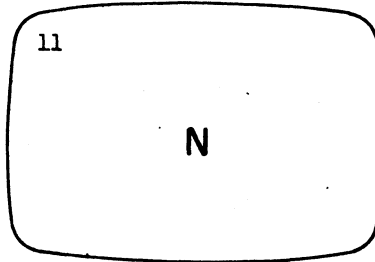


Figura 63

Caso a letra escolhida pelo TRS-2 tenha sido por exemplo o G = L\$(7) a primeira entrada sua é o 0 (zero) e a segunda é o 7 ou seja 07.

Como um trabalhinho instrutivo e em homenagem aos grandes recursos proporcionados pela subrotina que permite desenhar letras encha a tela do seu TRS-2 com a seguinte mensagem



# COPIANDO CONFORME O FIGURINO

Nós já vimos quase todos os recursos da "super-instrução" DRAW nos capítulos anteriores mas o melhor da instrução DRAW deixamos para os três últimos a contar deste.

Agora você vai aprender como se pode:

- 1) mudar o tamanho dos seus desenhos;
- 2) reservar memória para as páginas gráficas necessárias;
- 3) copiar um gráfico de uma página para outra;
- 4) usar tudo isto para se ter desenhos animados na tela;
- 5) utilizar a instrução PMODE para escolher a página gráfica;
- 6) usar muitas páginas uma em cada modo gráfico.

Para começar enfatizamos que o seu super micro TRS-2 além de colorido tem dentro do DRAW a possibilidade de poder ampliar ou reduzir qualquer figura que possa ser gerada pelo DRAW.

O comando geral para este fim (dentro da STRING após o DRAW) é

$S \times$  —————> um inteiro entre 1 e 62  
 (veja a Tabela 6)

↙  
 símbolo para  
 o modo escala

Valor de x	Escala na tela
1	1/4
2	2/4
3	3/4
4	4/4
⋮	⋮
60	60/4 = 15
61	61/4
62	62/4 = 15,5

Tabela 6

O comando  $S_x$  pode ser inserido na STRING após o DRAW onde for necessário.

Dentro de um dado programa todos comandos de movimento absoluto e relativo serão ampliados ou reduzidos de acordo com o último comando  $S_x$  que tenha sido executado.

Se nenhum comando  $S_x$  foi executado então é usado o comando  $S_4$  isto é  $4/4 = L$ .

Desta forma se escrevermos dentro de um programa:

```
DRAW "S8; EM 90,100; U25; L25; D25; R25"
```

as linhas retas serão desenhadas com 50 unidades para cima, 50 unidades para a esquerda, 50 unidades para baixo e 50 unidades para a direita embora dentro da instrução DRAW se tenha indicado apenas 25 unidades.

Tudo isto é devido a  $S_8$  pois  $8/4 = 2$  e consequentemente mudamos a escala para o dobro.

Não esqueça porém que para mudar um valor dentro da STRING após o DRAW você deve usar a função  $STR\$$  e isto vale também no caso particular de se querer mudar a escala.

Como uma aplicação deste modo vamos elaborar um programa que permita obter o que está mostrado na Figura 65 onde se amplia o lado do quadrado com um canto superior da esquerda

em (5,5) de 100%, 200%, 300%, ..., 600%.

Parece até a inflação na nossa querida pátria!?!?!?

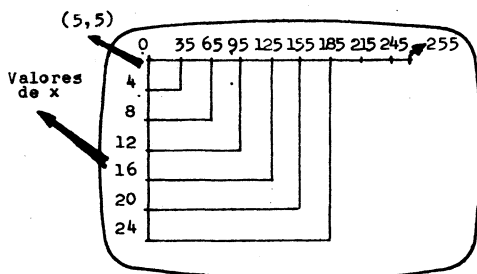


Figura 65



O.I. Não se afobe se o que sair não for exatamente um quadrado pois existe uma "distorsão" entre a altura e a largura. Você lembra da excentricidade no caso da circunferência?

Aí vai o programa:

```

10 PMODE 4,1
20 PCLS
30 SCREEN 1,1
40 REM É AGORA QUE VAMOS FAZER OS QUADRADOS, TODOS COM UM
   VÉRTICE EM 5,5
50 FOR J=4 TO 24 STEP 4
60 S$ = "S" + STR$(J) + ","
70 DRAW S$ + "EM 5,5; R30D30L30U30"
80 NEXT J
90 REM OLHE AGORA OS TEUS LINDOS QUADRADOS AUMENTADOS
100 GOTO 100

```

Que tal você agora como uma tarefa intelectual elaborar um programa que permita sentir todas as escalas possíveis aplicando isto a um pequeno quadrado de lado 10 cujo vértice (canto superior da esquerda) esteja no ponto 3,3?

Gostou da idéia?

Bem, use o programa acima como modelo para o seu ou então aprenda o que vem a seguir.

Mas lindo mesmo seria se você pudesse obter todos os seus quadrados concêntricos e centrados em (128,96) conforme mostrado na Figura 66.

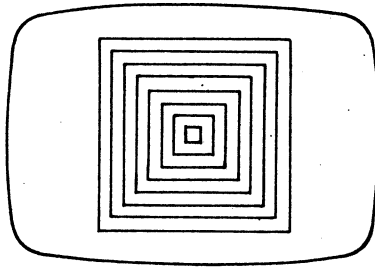


Figura 66

Teclé o seguinte programa e constate como isto é simples:

```

10 PMODE 4,1
20 PCLS
30 SCREEN 1,1
40 REM VAMOS VER TODAS AS AMPLIAÇÕES
50 FOR J=4 TO 62 STEP 8
60 A$ = "EM 128,96; BH 5"
70 S$ = "S" + STR$(J) + ";"
80 DRAW S$ + A$ + "R10D10L10U10"
90 NEXT J
100 GOTO 100

```

tem-se aqui um deslocamento sob um ângulo de 315°

Execute e veja se o que falamos realmente se realizou.

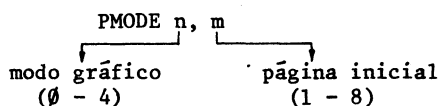


Com o que se mostrou no último programa fica bem simples resolver a "tarefinha" intelectual há pouco proposta.

#### VIRANDO AS PÁGINAS NO MODO $\emptyset$ (ZERO)

É hora agora de saber como se pode virar uma "página" e aprender como se distribui no seu TRS-2 a memória gráfica.

Você não esqueceu ainda porque a instrução PMODE tem dois parâmetros.



Os modos de alta resolução usam mais páginas de memória que os de baixa resolução como aliás está mostrado na Tabela 7 e nas Figuras 67a), b) e c).

Modo	Páginas da memória	Cores
PMODE 4	4	2
PMODE 3	4	4
PMODE 2	2	2
PMODE 1	2	4
PMODE $\emptyset$	1	2

Tabela 7

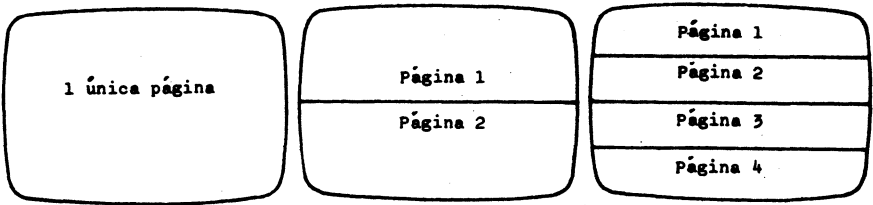
Quanto maior for o número de cores disponíveis ou quanto maior for a resolução mais memória é necessária.

Cada página gráfica usa aproximadamente 1,5 K de memória (aproximadamente 1500 posições).

É no PMODE 3 e 4 que se usa mais memória.

Normalmente quatro páginas de memória são reservadas para gráfico quando o TRS-2 está ligado.

Para se ter mais páginas do que estas basta executar a instrução.



a) PMODE 0  
Uma página enche a tela

b) PMODE 1 ou 2  
Duas páginas são necessárias para encher a tela

c) PMODE 3 ou 4  
Quatro páginas são necessárias para encher a tela

Figura 67

PCLEAR n

n pode ser um número entre 1 e 8

A medida que se vai reservando mais páginas a memória disponível do TRS-2 vai diminuindo como se mostra na Tabela 8.

n	memória disponível
1	13095
2	11559
3	10023
4	8487
5	6951
6	5415
7	3879
8	2343

Tabela 8 - Memória disponível após PCLEAR n

Vejamos agora como se podem virar as páginas e comecemos com o menor modo de resolução reservando 2 páginas de me

mória para os gráficos.

Vamos desenhar um retângulo na primeira página e uma circunferência na segunda página.

Não será usado neste programa a instrução SCREEN enquanto se fazem os desenhos.

A instrução SCREEN é usada neste programa para exibir o que você está desenhando.

```

5 REM VAMOS RESERVAR MEMÓRIA E ATIVAR A TELA
10 PCLEAR 2
20 PMODE 0,1
30 PCLS
40 REM VAMOS DESENHAR UM RETÂNGULO
50 DRAW "BM 80,50; R70D30L70U30"
60 REM AQUI TROCA-SE DE PÁGINA E DESENHA-SE A CIRCUNFERÊN
CIA
70 PMODE 0,2
80 PCLS
90 CIRCLE (128,96),65
100 REM AGORA É HORA DE CONFERIR E VIRAR AS PÁGINAS ONDE
ESTÁ CADA DESENHO
110 FOR P=1 TO 2
120 PMODE 0,P
130 SCREEN 1,0 —————> agora exibe-se a página
140 FOR K=1 TO 299: NEXT K
150 NEXT P
160 GOTO 110

```

Execute o programa e veja que você terá alternadamente na tela o que se exibe nas Figuras 68a) e b).

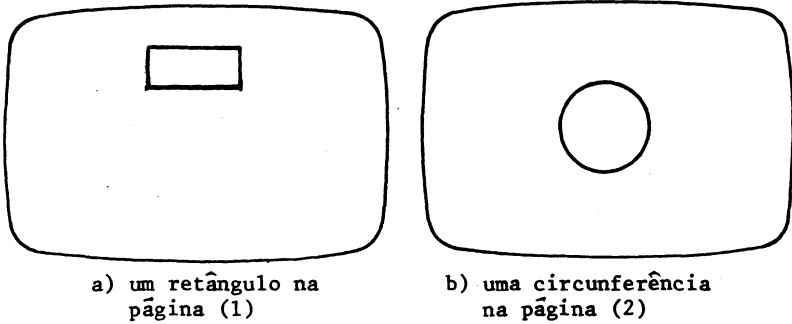


Figura 68

Como uma outra tarefa para agitar mais ainda a sua massa cinzenta elabore um programa que permita exibir uma circunferência de círculo de raio 35 em diferentes posições em cada.

O assunto agora é sobre como se pode "colar" ou muito melhor copiar o gráfico de uma página para outra.

Para fazer isto devemos usar a instrução

PCOPY X TO Y

↙ copiar da página X

↘ para a página Y

Na instrução P , X e Y podem ser inteiros de 1 a 8.

Com o auxílio da instrução PCOPY pode-se produzir um movimento aparente na tela do vídeo.

Suponhamos que você queira mover uma circunferência de um certo ponto para outro de um quadrado.

Isto pode ser feito no sentido anti-horário (Figura 69).

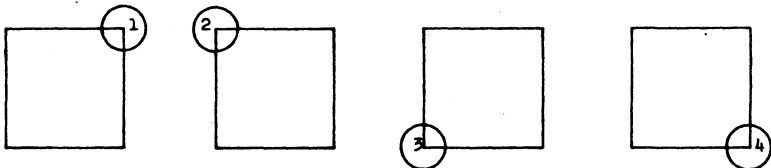


Figura 69 - Movimento anti-horário

ou no sentido horário (Figura 70).

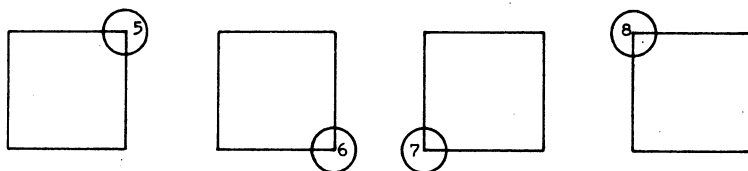


Figura 70 - Movimento horário

Caso você esteja no PMODE 0 pode então usar uma página da memória gráfica para cada posição da circunferência.

O quadrado aparecerá sempre na mesma posição em todas as páginas, porém a posição da circunferência muda.

Dá também para perceber que algumas das posições das circunferências são as mesmas; é o caso das posições 1 e 5, 2 e 8, 3 e 7 e 4 e 6.

A oportunidade ideal para usar a instrução vem a seguir!!!

Inicialmente vamos desenhar um quadrado na 1ª página gráfica e copiar a mesma nas outras sete páginas.

Após isto vamos colocar as circunferências nas posições corretas e copiar nas páginas apropriadas.

Finalmente na 3ª parte do programa vamos ativar a exibição e virar as páginas.

Inicialmente usaremos uma pausa relativamente grande e em seguida vamos diminuir esta pausa e com isto a idéia do movimento no sentido anti-horário e horário torna-se bem delineada no programa que chamaremos de "MEXENDO-SE".

```
5 REM ESTE PROGRAMA CHAMA-SE **MEXENDO-SE**
10 REM VAMOS LIMPAR AS 8 PÁGINAS E DESENHAR UM QUADRADO
20 PCLEAR 8
30 PMODE 0,1
```

```

40 PCLS
50 DRAW "EM 100,80; R40D40L40U40"
60 REM VAMOS COPIAR ESTE QUADRADO NAS OUTRAS 7 PÁGINAS
70 FOR P=2 TO 8
80 PCOPY 1 TO P
90 NEXT P
100 REM CIRCUNFERÊNCIA DA PÁGINA 1 PARA A 5
110 PMODE 0,1
120 CIRCLE (140,80),13
130 PCOPY 1 TO 5
140 REM CIRCUNFERÊNCIA DA PÁGINA 2 PARA A 8
150 PMODE 0,2
160 CIRCLE (100,80),13
170 PCOPY 2 TO 8
180 REM CIRCUNFERÊNCIA DA PÁGINA 3 PARA A 7
190 PMODE 0,3
200 CIRCLE (100, 120),13
210 PCOPY 3 TO 7
220 REM CIRCUNFERÊNCIA DA PÁGINA 4 PARA A 6
230 PMODE 0,4
240 CIRCLE (140,120),13
245 PCOPY 4 TO 6
250 REM É AGORA QUE VÃO SER EXIBIDAS AS PÁGINAS EM SUCESSÃO
360 FOR P=1 TO 8
370 PMODE 0,P —————> aqui movimenta-se as páginas
380 SCREEN 1,0 —————> aqui "liga-se" a tela
390 FOR K=1 TO 299: NEXT K ———> (mude depois esta instrução
para
400 NEXT P                               290 FOR K=1 TO 30: NEXT K
410 GOTO 360

```

Se você gostou do que aconteceu até agora vai ficar com o coração palpitante de tanta emoção...

Vamos produzir dois movimentos.

Aliás, enquanto isto não ocorre tente, para se antecipar a emoção e já estar preparado(a) para a mesma, alisar a sua "cuquinha" em um movimento circular com a mão esquerda e com a direita faça um movimento circular sobre a sua mão saliente "barriguinha".

Ah! Ah! Ah!

Os movimentos circulares devem ser em sentidos opostos pois senão perde a graça...

Veja se consegue ficar neste estado "acariciante" pelo menos 1 minuto.

O que, você não consegue?

Que falta de coordenação !!!!

Bem, nisto o seu TRS-2 talvez seja melhor que você.

Vamos acrescentar ao programa "MEXENDO-SE" um retângulo e em cada um dos seus vértices e nos pontos médios dos lados vamos colocar 8 circunferências (veja a Figura 71).

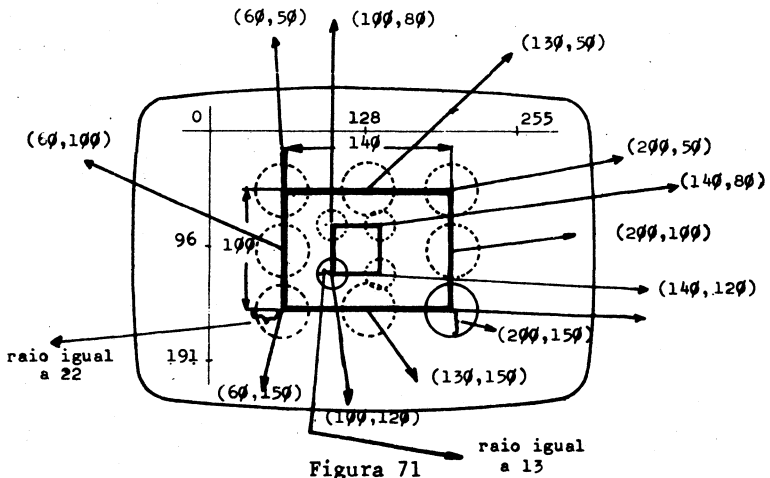


Figura 71

Inicialmente você deve desenhar o retângulo na página 1 e para tanto entre com a instrução.

55 DRAW "EM 60,50; R140;D100; L 140; U 100"

Em seguida devem ser colocadas as circunferências em oito posições diferentes.

260 PMODE 0,1: CIRCLE (200,50),22  
 265 PMODE 0,2: CIRCLE (130,50),22  
 270 PMODE 0,3: CIRCLE (60,50),22  
 275 PMODE 0,4: CIRCLE (60,100),22  
 280 PMODE 0,5: CIRCLE (60,150),22  
 285 PMODE 0,6: CIRCLE (130,150),22  
 290 PMODE 0,7: CIRCLE (200,150),22  
 295 PMODE 0,8: CIRCLE (200,100),22

Execute agora e veja que você tem uma circunferência que se movimenta em torno de um quadrado ora em um e ora em outro sentido e além disto uma outra circunferência que se movimenta no sentido anti-horário em torno de um retângulo.

Caso você queira mudar o sentido do movimento basta modificar a instrução 360 para:

360 FOR P=8 TO 1 STEP -1

Faça isto, execute e veja como as "coisas" se invertem.

Não tem ninguém (humano) com três mãos aqui na Terra mas você já viu muito malabarista virar pratinho com uma mão, argola com um pé e ainda com a cabeça dar sensacionais "toques" numa bola...

Porém eu disse que uma criatura deste tipo é um(a) malabarista e você pode não ser, mas o seu TRS-2 pode ser!!!!

Como agora está na moda uma "propaganda" baseada na lei de Newton da ação e reação onde se mostra aos (às) "telemaniacos(as)" o que uma inocente "maçanzinha" despencando do 5º andar pode fazer ao chocar-se com a capota de um carro vamos, sem nenhuma destruição, acrescentar duas "bolas" que vão caindo dos últimos andares de dois prédios diferentes mas de mesma altura.



Para tanto faça o seguinte acréscimo final ao seu programa "MEXENDO-SE"

```

300 PMODE 0,1: CIRCLE (20,10),6: CIRCLE (220,10),6
305 PMODE 0,2: CIRCLE (20,30),6: CIRCLE (220,30),6
310 PMODE 0,3: CIRCLE (20,60),6: CIRCLE (220,60),6
315 PMODE 0,4: CIRCLE (20,90),6: CIRCLE (220,90),6
320 PMODE 0,5: CIRCLE (20,120),6: CIRCLE (220,120),6
325 PMODE 0,6: CIRCLE (20,150),6: CIRCLE (220,150),6
330 PMODE 0,7: CIRCLE (20,170),6: CIRCLE (220,170),6
335 PMODE 0,8: CIRCLE (20,184),6: CIRCLE (220,184),6

```

Entre com estas últimas modificações e execute este longo, porém instrutivo programa.

Você consegue acompanhar todos os movimentos que ocorrem ao mesmo tempo?

O seu TRS-2 parece até um artista de circo, não é?

Bem, não sei se você vai ser capaz, mas acho que é minha obrigação sugerir que com o que lhe foi ensinado tente elaborar um programa que faça uma pequena circunferência de raio 7 unidades "rodar" em torno de uma circunferência maior de raio 65 unidades como está mostrado na Figura 72.

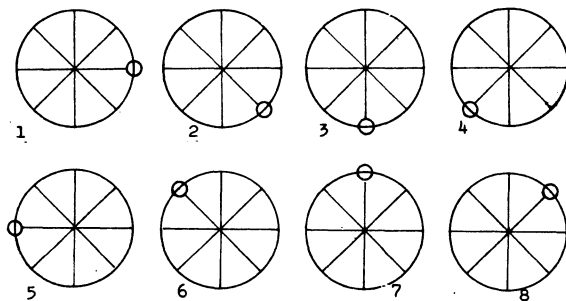


Figura 72



# PAGINAS USADAS NA TOTALIDADE

Vamos usar e abusar das páginas gráficas do TRS-2 e para tanto neste programa exibiremos quadrados de tamanho ca da vez maior a medida que se viram as páginas gráficas.

Um quadrado é desenhado tendo no seu interior um círculo pintado.

Os círculos são pintados com cores diferentes em cada par de páginas.

Vamos usar o Modo 1 e portanto 2 páginas são necessárias para se encher a tela.

O número da página é usado para determinar o movimento relativo do canto superior e o tamanho do quadrado, assim como o raio e a cor da "tinta" que é usada para pintar o círculo.

Aí está o programa

```

10 PCLEAR 8 —————> oito páginas são necessárias no programa
20 FOR P=1 TO 8 STEP 2
30 PMODE 1,P: PCLS —————> limpa todas as páginas
40 NEXT P
50 PMODE 1,1
60 DRAW "BM 80,50" —————> fixa-se um ponto inicial para o quadrado
70 FOR P=1 TO 8 STEP 2 —————> desenham-se todos os quadrados e
                                > circunferências
80 PMODE 1,P
90 A$ = STR$(2+P): R = 7*P: C = (P+3)/2

```

```

100 DRAW "S" + A$ + "BM -" + A$ + ", -" + A$ + "R48D48L48U48"
110 CIRCLE (100,70),R
120 PAINT (100,70),C,4
130 NEXT P
140 FOR P=1 TO 8 STEP 2 —————> exibe-se duas páginas em cada
                                     instante
150 PMODE 1,P
160 SCREEN 1,0
170 FOR K=1 TO 599: NEXT K
180 NEXT P
190 GOTO 140

```

### Explicações

Oito páginas vão ser limpas para o gráfico na linha 10.

As páginas são limpas com o auxílio do laço FOR-NEXT (linhas 20 até 40).

Lembre que o Modo 1 usa duas páginas para encher a tela (Tabela 7).

É na linha 60 que se escolhe a posição inicial do quadrado e coloca-se a mesma no ponto (80,50) da tela (isto é um pouco para a esquerda e um pouco acima do centro).

As linhas 70 até 130 permitem desenhar o quadrado e pintar um círculo em cada uma das duas páginas.

O quadrado está em uma escala que varia de acordo com a página escolhida (veja a Tabela 9).

A circunferência está no centro do quadrado e está pintada com uma cor que depende da página escolhida.

Todos os desenhos são feitos antes que a tela esteja ligada.

O laço FOR-NEXT que começa na linha 140 e vai até a linha 180 exibe as telas na ordem indicada na Tabela 9.

Página	Circunferência		Quadrado	
	Raio	Cor	Escala	Lado
1	8	2	1/4	12
3	24	3	3/4	36
5	40	4	5/4	60
7	56	5 ou 1	7/4	80

Tabela 9

Na linha 170 foi introduzida uma pausa para que você possa admirar as diversas saídas.

E acabou o Capítulo 24.

Também você já está começando a ficar cansado e também ficar muito tempo num capítulo de número 24 pode não ser totalmente uma boa...

Mas a razão maior é permitir que você chegue logo ao último capítulo e se torne um(a) "expert" no que se refere a gráficos no TRS-2 !!



# CANHAO ROTATIVO

Falta sô mais uma "coisinha" para você ter o domínio da super instrução DRAW na sua plenitude.

Estou me referindo ao modo ângulo que se ativa com o comando A seguido de um número.

A forma geral é Ax onde x é o código do ângulo (de 0 até 3).

Todos os ângulos são medidos no sentido horário.

Depois que a opção A é incluída no comando DRAW, todas as linhas subsequentes serão desenhadas com um deslocamento angular que se informou com o Ax.

No caso de se ter:

$$\left\{ \begin{array}{l} x = 0 \text{ tem-se } 0^\circ; \\ x = 1 \text{ tem-se } 90^\circ; \\ x = 2 \text{ tem-se } 180^\circ; \\ x = 3 \text{ tem-se } 270^\circ. \end{array} \right.$$

Caso o ângulo não seja especificado, o TRS-2 usa por conta própria A0.

Para se ilustrar o uso desta opção vamos apresentar um programa que simula o cano de um canhão que é desenhado sucessivamente e depois é apagado a medida que roda de  $45^\circ$  em  $45^\circ$ .

Fizemos também o uso e abuso da operação de concatenação dentro da instrução DRAW.

Aí está o penúltimo programa

```

10 PMODE 3,1
20 PCLS
30 SCREEN 1,1
40 REM É AQUI QUE SE AJUSTAM OS MOVIMENTOS BÁSICOS
50 B$ = "BM 128,96; U80"
60 I$ = "BM 128,96; E60"
70 REM É AGORA QUE VAMOS ATRIBUIR VALORES AOS ÂNGULOS
80 FOR J=0 TO 3
90 A$(J) = "A" + STR$(J)
100 NEXT J
110 REM VAMOS DESENHAR UMA PEQUENA CIRCUNFERÊNCIA NO CENTRO
120 CIRCLE (128,96),13,7
130 REM É AGORA QUE SE DESENHA E APAGA O CANO DO CANHÃO
140 FOR I=0 TO 3
150 DRAW A$(I) + "C8" + B$ → desenha-se uma posição horizon-
    tal (ou vertical)
160 DRAW A$(I) + "C5" + B$ → apaga-se esta última posição
170 DRAW A$(I) + "C8" + I$ → desenha-se uma posição "incli-
    nada"
180 DRAW A$(I) + "C5" + I$ → apaga-se a última posição
    inclinada
190 NEXT I
200 GOTO 140

```

Execute e veja se você vê na tela o que se mostra na Figura 73.

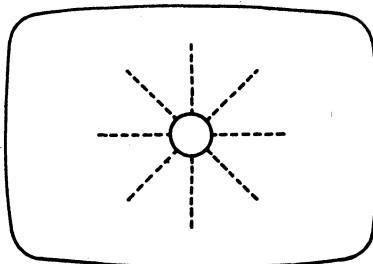


Figura 73



Neste último programa que vamos apresentar você obtém o recurso para elaborar um desenho com comandos individuais.

O seu TRS-2 permite aumentar o seu desenho da forma que quiser.

O TRS-2 lhe perguntará a cada etapa sobre o movimento que você deseja efetuar.

O seu programa começa com a "caneta invisível" que desenha, na posição central da tela.

Aí o TRS-2 lhe perguntará se você quer a caneta levantada ou abaixada.

Se a sua escolha for caneta levantada, o programa permite que você se desloque com a caneta em *movimento relativo* para qualquer posição da tela desejada *sem desenhar*.

Se por outro lado a sua escolha for caneta abaixada então o seu desenho irá começar a partir do centro da tela.

Aí está o "the last"

```

10 K=0
20 CLS: CLEAR
30 INPUT "CANETA ABAIXADA (S/N)?";R$
40 CLS: S$ = "": D$=""
50 IF R$ = "S" THEN GOSUB 1000 ELSE GOSUB 2000
60 A$ = A$ + S$ + D$
70 INPUT "DE QUE MAGNITUDE E O SEU DESLOCAMENTO?";M
80 CLS
90 A$ = A$ + STR$(M)
100 K = K+1
110 PMODE 4,1
120 IF K=1 THEN PCLS
130 SCREEN 1,1
140 DRAW "BM 128,96" + A$

```

É aqui se se escolhe para se desenhar ou para se mover "em branco"

com esta instrução limpa-se as STRINGs antigas

limpa a tela se é o primeiro movimento

começa no centro e acrescenta A\$

```

150 W$ = INKEY$: IF W$ = "" THEN 150 → espera pelo próximo
                               movimento
160 GOTO 30
1000 CLS: INPUT "COM A CANETA ABAIXADA QUERO A DIREÇÃO";D$
1010 CLS: RETURN
2000 CLS: INPUT "COM A CANETA LEVANTADA QUAL É O DESLOCAMEN-
      TO PARA X (ABSCISSA)?";X$
2010 CLS: S$ = "BM" + X$ + ", "
2020 INPUT "COM A CANETA LEVANTADA QUAL É O DESLOCAMENTO
      PARA Y (ORDENADA)?";Y$
2030 CLS: S$ = S$ + Y$
2040 GOSUB 1000
2050 RETURN

```

#### Explicações complementares sobre o programa

O programa começa no modo texto.

A linha 30 permite a você decidir se quer um movimento em branco com caneta para cima (ou seja levantada) ou um movimento desenhando (caneta abaixada).

A subrotina 1000 é a escolhida com a caneta abaixada e aí você deve entrar com R, L, U ou D.

Para entrar na subrotina 1000 você deve responder "S" na linha 30.

Caso você não entre com "S" na linha 30 irá, devido a linha 50, para a subrotina que começa em 2000.

A subrotina 2000, permite que se faça um movimento "em branco".

Esteja certo de que usou um sinal mais ("+") ou um sinal ("-") na frente do movimento na direção Ox.

Se o movimento na direção Oy é negativo o sinal menos ("-") precisa também ser incluído.

O TRS-2 sai então da subrotina que começa em 2000, sabendo segundo que direção a caneta deve se mover sem desenhando.

Isto tudo é concatenado nas linhas 2010 e 2030.

Voltando ao programa principal ocorrem ainda concatenações muito importantes nas linhas 60 e 90.

A linha 100 define se este é o primeiro movimento.

É nas linhas 110 até 140 que se prepara a execução e exibição do desenho.

Na linha 120 se determina que a tela gráfica precisa ser limpa (sõmente no primeiro movimento) ou se um conjunto prêvio de movimentos foi deixado na tela gráfica.

Lembre que o TRS-2 começa sempre no centro da tela assim que o primeiro movimento é feito, inclua então um movimento em vazio caso não queira começar o seu desenho no centro da tela.

Veja se consegue desenhar o que está mostrado na Figura 74 ou seja uma espirolateral toda irregular.

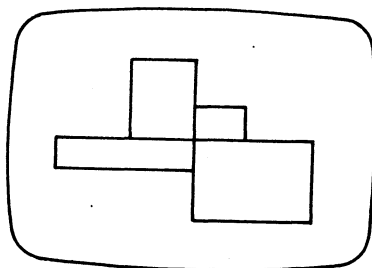


Figura 74

Tente que você conseguirá após uma pequena canseira.

Isto permitirá que lhe diga

**PARABÉNS!!**

por ter conseguido o desenho e principalmente por ter estudado, compreendido e executado todos os programas apresentados neste livro.

O.I. FINAL

Não esqueça que toda livraria que se preze tem outros livros da série TRS-2 para vo cê adquirir o domínio, com o seu supermi-cro, de outras áreas.

Adquira-os!!!

# Apêndice 1

## Mensagens de erro no CBE (COLOR BASIC ESTENDIDO)

- /Ø** Divisão por zero. Foi pedido ao TRS-2 para dividir um número por zero, o que é impossível.
- AO** Você quer abrir um arquivo de dados que já foi aberto. Se você apertar a tecla RESET durante a operação receberá esta mensagem.  
Desligue o TRS-2 e ligue de novo.
- BS** Índice inválido. Os índices numa matriz estão fora do intervalo. Use DIM para dimensionar a matriz. Por exemplo, se você tiver A(23) em seu programa, sem uma instrução DIM que dimensiona a matriz A para 23 elementos ou mais verá aparecer essa mensagem de erro.
- CN** Não pode continuar. Se você usar um comando CONT e está no fim de um programa ou em alguma outra condição que não dá para prosseguir terá essa mensagem de erro.
- DD** Tentativa de redimensionar uma matriz.  
Uma matriz pode ser dimensionada uma única vez. Por exemplo, você não pode ter DIM B(15) e DIM B(27) no mesmo programa.
- DN** Erro no número de periférico. Somente três números para periféricos podem ser usados com OPEN, CLOSE, PRINT ou INPUT: Ø, -1, ou -2. Se você usar outro número terá essa mensagem de erro.

- DS** Comando direto. Há um comando direto num arquivo de dados. Isto pode acontecer se você tentar carregar um arquivo de dados em fita.
- FC** Chamada de função ilegal. Isto ocorre se você usar um parâmetro (número ou variável) inválido ou que está fora do intervalo. Por exemplo `PLAY":"` causará este tipo de erro.
- FD** Dados de arquivo inválido. Este erro acontece quando você imprime dados num arquivo ou introduz dados de um arquivo, utilizando o tipo errado de variável para o dado em questão. Por exemplo, `INPUT -1,X`, quando o dado de um arquivo é uma `STRING`, causa este tipo de erro.
- FM** Modo de arquivo inválido. Isto ocorre quando você quer introduzir dados de um arquivo aberto apenas para saída ou imprimir dados de um arquivo aberto para entrada.
- ID** Comando direto ilegal. Você só pode usar `INPUT` como uma linha de programa não como uma linha de comando.
- IE** Entrada de valores depois do final de arquivo. Use `EOF` para verificar quando você chegar no final do arquivo. Quando você tiver chegado use `CLOSE` e feche-o.
- IO** Erro de entrada/saída. Várias vezes isto é provocado quando se tenta entrar com um programa ou arquivo de dados de uma fita ruim.
- LS** `STRING` muito longa. A `STRING` só pode ter 255 caracteres no máximo.

- NF** NEXT sem FOR. NEXT está sendo usado sem o comando FOR correspondente. Isto também ocorre quando você tiver as linhas NEXT invertidas em LOOPS internos ou seja aninhados.
- NO** Arquivo não aberto. Você não pode operar um arquivo sem ter antes aberto o mesmo.
- OD** Há carência de dados. Um READ foi executado com dados insuficientes. Um ou mais comandos DATA podem ter sido deixados fora do programa.
- OM** Há falta de memória. Toda a memória disponível já foi utilizada.
- OS** Não há espaço suficiente na memória para as operações com STRINGS. Use CLEAR no começo do seu programa para reservar mais espaço para as STRINGS.
- OV** Exceder. O número é muito grande para o TRS-2 manipular o mesmo.
- RG** RETURN sem GOSUB. Existe uma instrução RETURN em seu programa sem uma linha GOSUB correspondente.
- SN** Erro de sintaxe. Isso pode resultar de um comando escrito de forma errada, pontuação incorreta, parênteses abertos e não fechados ou um caractere ilegal. Digite a linha de programa ou o comando novamente.
- ST** Fórmula de STRING muito complexa. A operação com a STRING era muito complicada para se operar. Divida a operação em etapas menores.

TM

Tipo errado. Isso ocorre quando você tenta atribuir dados numéricos a uma variável STRING. Por exemplo X\$ = 13 ou então dados alfanuméricos a uma variável numérica como por exemplo A = TRS-2

UL

Linha indefinida. O programa possui um GOTO ou um GOSUB, em alguma outra linha de desvio ordenando que se vá a uma linha inexistente no programa.



# Apêndice 2

## Sumário do CBE (COLOR BASIC ESTENDIDO)

ABS

Computa o valor absoluto de um número ou expressão numérica.

ASC

Converte o primeiro caractere de uma STRING para o seu código ASCII correspondente. Por Exemplo ASC ("VICTOR") converte "V" para o seu código ASCII que é 86.

Uma lista de códigos ASCII está na Tabela 3.

ATN

Dá o arcotangente em radianos.

AUDIO

Liga ou desliga o som de seu gravador para a T.V.

CHR\$

Converte um código ASCII para o seu caractere correspondente. Uma lista completa de códigos está na Tabela 3.

CIRCLE

Desenha uma circunferência com centro no ponto (X,Y), raio R, cor especificada por C e razão altura/largura ou seja a excentricidade indicada por E. A circunferência pode começar (I) e terminar (F) em pontos especificados.

A forma geral da instrução é

C(X,Y),R,C,E,I,F

**CLEAR**

Reserva espaço na memória do TRS-2 para trabalho com STRINGS. Sem CLEAR o computador reserva automaticamente 200 caracteres.

Se você está chamando um programa em linguagem de máquina você pode usar um segundo número para especificar o mais alto endereço que o BASIC pode usar. CLEAR sempre zera todas as variáveis e torna as STRINGS nulas.

**CLOAD**

Chama o primeiro programa de uma fita cassete. Você pode especificar o nome do programa.

**CLOADM**

Carrega programa em linguagem de máquina do gravador. Você pode mencionar um endereço de desvio para adicionar ao endereço carregado.

**CLOSE**

Fecha o acesso a um arquivo especificado.

Se você não especificar o dispositivo todos os arquivos abertos são fechados.

**CLS C**

Limpa a tela para o código de cor (C) que você especificar. Veja na Tabela 1 a lista de códigos de cor.

**COLOR**

Permite ajustar as cores do fundo e do primeiro plano.

**CONT**

Continua executando o programa após apertar o BREAK ou usar o STOP.

**COS**

Dá o cosseno de um ângulo medido em radianos.

**CSAVE**

Guarda um programa na fita cassete. Você deve usar um nome para o programa com não mais de oito caracteres/espaço.

**CSAVEM**

Grava um arquivo em linguagem de máquina.

**DATA**

Guarda dados do seu programa. Use READ para acessar estes dados e atribuir as variáveis.

**DEF FN**

Define a função numérica.

**DEFUSRn**

Define o ponto de entrada para a função USR. O n pode tomar valores entre 0 e 9.

**DEL**

Possibilita a eliminação de linhas do programa. Com o DEL anula-se o programa todo; com DELn anula-se a linha n; com DELn - anula-se todas as linhas a partir de n; com o DEL-n anula-se toda todas as linhas até a linha n e com DEL n<sub>1</sub>-n<sub>2</sub> anulam-se as linhas entre as linhas n<sub>1</sub> e n<sub>2</sub>.

**DIM**

Reserva espaço na memória para as matrizes que você especificar.

As matrizes podem ser numéricas ou STRING (alfa numéricas).

**DLOAD**

Carrega um programa em C B E em uma definida velocidade (0 = 3000 band; 1 = 12000 band).

**DRAW**

É a mais poderosa instrução gráfica e permite desenhar uma linha reta com ponto de partida, comprimento e cor previamente definidos.

**EDIT**

Permite a edição de uma linha do seu programa. Para saber todos os recursos de edição leia o manual do TRS-2

**END**

Indica o fim de um programa.

EOF

Verifica se você chegou ao fim de um arquivo de dados. Se você tiver EOF(-1) será verdade que não há mais dados em caso contrário tem-se EOF(0).

EXEC

Transfere controle para programas em linguagem de máquina para o endereço que você especificar. Se o endereço for omitido o TRS-2 irá usar um endereço especificado no último comando CLOADM.

EXP

Permite obter o valor de uma função exponencial de base e ( $e=2,7182818$ ) isto é da função  $e^x$ .

FIX

Toma apenas a parte inteira de um número.

FOR - TO  
STEP/  
NEXT

Cria um laço (ciclo, malha ou LOOP) em seu programa que o TRS-2 deverá repetir do primeiro ao último número que você definir.

Você deve usar STEP para especificar o passo ou seja o incremento ou decremento do número a cada volta completada. Se o STEP for omitido o incremento será um.

GET

Permite ler o conteúdo gráfico de um retângulo em uma matriz para posterior utilização na instrução PUT.

GOSUB

Envia o TRS-2 para uma subrotina que começa no número de linha que você especificar após o GOSUB

GOTO

Desvia o TRS-2 para o número de linha que você especificar após o GOTO.

HEX\$

Calcular o valor hexadecimal.

IF/THEN  
ELSE

Testa as relações. Se for verdadeira o TRS-2 executa a instrução seguinte ao THEN. Se não for verdadeira o TRS-2 executa a instrução seguinte ao ELSE ou se não existir ELSE vai para a próxima linha do programa.

INKEY\$

Verifica que tecla foi pressionada no teclado.

INPUT

O TRS-2 para e espera a entrada de dados através do teclado.

INPUT#-1

Faz com que o TRS-2 pare e espere uma entrada de dados do gravador.

INSTR

Procura pela primeira ocorrência da STRING B\$ na STRING A\$ e volta à posição na qual ela aconteceu.

INT

Transforma um número atribuído a uma variável no maior inteiro imediatamente menor que o valor numérica desta variável.

JOYSTIC

Permite ler a coordenada horizontal ou vertical do "joystick" da esquerda ou da direita.

JOYSTK(0) — retorna a posição horizontal do "joystick" da esquerda

JOYSTK(1) — retorna a posição vertical do "joystick" da esquerda

JOYSTK(2) — retorna a posição horizontal do "joystick" da direita

JOYSTK(3) — retorna a posição vertical do "joystick" da direita.

LEFT\$

Toma a parte esquerda de uma STRING. Você especifica a STRING e o comprimento da parte esquerda.

**LEN**

Fornecer o comprimento de uma STRING.

**LET**

Atribuir valores a variáveis.

Totalmente desnecessária no CBE (COLOR BASIC estendido).

**LIST**

Lista o programa todo ou as linhas do programa que você especificar.

Cuidado, pois pode não caber tudo na tela!!!

**LLIST**

Lista o programa ou as linhas de programa que você especificar na impressora.

**LINE**Permite desenhar uma linha de um ponto inicial  $(X_1, Y_1)$  a um ponto final  $(X_2, Y_2)$ .

Caso o ponto inicial seja omitido o centro da tela (128,96) ou o último ponto final é utilizado. O PSET após essas definições seleciona a cor do primeiro plano e o PRESET escolhe a cor do fundo. Usando a opção B desenha-se uma moldura usando-se como vértices opostos os pontos inicial e final.

A opção BF permite pintar o que está dentro da moldura retangular com a cor escolhida em PSET ou seja a do primeiro plano.

**LINE INPUT**

Permite a entrada de linhas de textos pelo teclado.

**LOG**

Com esta função calcula-se o logaritmo natural (base e) de uma variável numérica positiva.

**MEM**

Diz a você quanto espaço o TRS-2 ainda tem na memória.

**MID\$**

Fornece uma subSTRING de outra STRING. Você especifica a STRING a posição que começa a subSTRING e o seu comprimento. Por exemplo MID\$("NOMES",2,3) se transforma em OME.

**MOTOR**

Liga ou desliga o gravador.

**NEW**

Apaga tudo na memória.

**ON-GOSUB**

Envia o TRS-2 para a subrotina que você indicar.

**N -GOTO**

Desvia o TRS-2 para o número de linha que você indicar.

**OPEN**

Abre um arquivo especificado para a transmissão de dados para um periférico (dispositivo) especificado. A transmissão pode ser de entrada(I) ou saída(O).

Os periféricos são:

$$\left\{ \begin{array}{l} \# 0 = \text{tela} \\ \# -1 = \text{gravador} \\ \# -2 = \text{impressora} \end{array} \right.$$

não é necessário "abrir" a comunicação quando você está entrando com dados do teclado ou imprimindo na tela.

**PAINT**

Permite pintar gráficos na tela começando em um ponto especificado (X,Y) com a cor definida pelo código (C) e parando numa borda com cor especificada por (B).

**PCLEAR**

Permite reservar n números de páginas de memória gráfica.

**PCLS C**

Limpa a tela com a cor especificada no código(C).  
Caso o código seja omitido é usada a cor do fundo

**PCOPY**

Permite copiar gráficos de uma página (origem) para outra página (destino).

**PEEK**

Devolve o conteúdo da localização da memória do endereço especificado.

**PLAY**

Soa uma determinada nota (A-G ou 1-12) oitava(0), volume (V), comprimento da nota (L), ritmo (T), pausa (P) e possibilita ainda a execução de sub-STRINGS, assim como inclui o sustenido (# ou +) ou o bemol (-).

**PMODE**

Permite escolher o tipo de resolução gráfica e a página da memória para iniciar.

**POINT**

Diz se o ponto do qual você especificou as posições vertical e horizontal está ativado ou apagado. Retorna para -1 se o ponto estiver no modo caractere, 0 se não estiver ativado ou o código de cor se estiver ativado.

**POKE**

Coloca um valor na posição da memória que você especificar.

O valor, pode ser um número entre 0 e 255.

**POS**

Fornece a posição atual do cursor.

**PPOINT**

Testa se o ponto gráfico definido está ativado ou apagado e retorna o código da cor caso esteja aceso aquele ponto.

**PRINT**

Imprime a mensagem que você quer no periférico que você especificar. Se não houver especifica-



ção alguma sua mensagem será impressa na tela do vídeo.

**PRINT#-1**

Grava os dados na fita.

**PRINT#-2**

Imprime um item ou uma lista de itens na impressora.

**PRINT TAB**

Desloca o cursor para uma coluna especificada pelo argumento da função TAB e imprime.

**PRINT USING**

Imprime números em um formato previamente especificado.

Entre outras coisas possibilita introduzir o ponto decimal, coloca a vírgula a cada três números e pode-se preencher com asteriscos os espaços em branco na frente do número. Caso se queira pode ser incluído um cifrão (\$) antes do número.

**PRINT @**

Imprime a sua mensagem na posição que você especificar na tela de texto.

**PSET**

Coloca um ponto especificado na tela na cor definida. Caso o código de cor seja omitido é usada a cor do primeiro plano.

**PUT**

Guarda gráficos de uma matriz na tela.

O tamanho da matriz retangular deve ser igual ao tamanho retangular de GET.

**READ**

Lê o próximo item numa linha de dados DATA e atribui o mesmo à variável especificada.

**REM**

Permite a você inserir um comentário num programa. O TRS-2 ignora tudo que esteja numa linha de REM.

- RENUM** Permite a você remunerar as linhas do programa.
- RESET** Apaga um elemento de imagem da tela.
- RESTORE** Volta o programa para o primeiro item da primeira linha de (DATA)
- RETURN** Retorna o TRS-2 de uma subrotina de um programa em C B E para a instrução seguinte ao GOSUB.
- RIGHT\$** Toma a posição à direita de uma STRING começando na posição que você especificar.
- RND** Fornece um número pseudo-aleatório ou ao acaso dentro da faixa de 1 ao número que você especificar e que deve ser maior que 1 obviamente.
- RUN** Executa um programa desde o início ou desde a linha que você especificar.
- SCREEN** Permite a você escolher tela de gráficos (1) ou de texto (0) e o conjunto de cores (0 ou 1).
- SET** Desenha um ponto na tela na cor e localização que você especificar.
- SGN** Esta função dá o sinal de um número. Toma o valor 1 se o número é positivo, 0 se é zero e -1 se é negativo.
- SIN** Permite obter o seno de um ângulo dado em radianos.
- SKIPF** Escapa para o fim do próximo programa da fita ou para o fim do programa que você especificar.
- SOUND** Som no tom e na duração que você especificar. Ambos o tom e a duração podem ser um número entre 1 e 255

STOP	Faz o TRS-2 parar a execução de um programa.
STR\$	Converte um número para uma STRING.
SQR	Permite obter a raiz quadrada de um número real positivo.
TAB	Tabula na posição que você desejar (veja PRINT TAB)
TAN	Permite obter a tangente trigonométrica de um ângulo dado em radianos.
TIMER	Permite conhecer o conteúdo de um "cronômetro" ou então possibilita o seu ajuste.
TROFF	Desativa o traçador de um programa.
TRON	Ativa o traçador de um programa.
USRn	Chama uma subrotina em linguagem de máquina.
VAL	Converte uma STRING em um número.
VARPTR	Faz voltar o apontador de endereço para uma variável especificada.



## OUTRAS OBRAS DO AUTOR

Caderno de Elementos de Computação (linguagem FORTRAN)  
Linguagem BASIC  
BASIC sem segredos  
TK-Divertindo  
TK-Lembrando  
TK-Calculando  
Conhecendo e Utilizando o TK-2000  
TK-2000 na Matemática  
Dê um APPLE à sua vida  
Imprimindo maravilhas com a GRAFIX  
Brincando com o TRS Color  
BASIC no TK 90X

## PRÓXIMOS LANÇAMENTOS

Jogos e Desenhos no TK 90X  
Gráficos no TK 90X  
TK-2000 nos Gráficos e na Música  
TK-2000 nas Ciências Humanas

**N.º 1489**

**IMPRESSO NAS OFICINAS DE EDIÇÕES LOYOLA  
RUA 1822 N.º 347 — FONE: 914-1922 — SP**



Este livro permite desenvolver sua criatividade e imaginação de forma concreta, definida e colorida, capacitando-o (a) a explorar toda gama de recursos gráficos do Basic através do microcomputador TRS-80 COLOR ou dos compatíveis nacionais, tais como o CP-400, COLOR 64 etc.

Programas para conjuntos de retas, circunferências, molduras com o interior pintado, desenhos sofisticados, pequenos jogos. A combinação de tudo isto você irá encontrar neste livro.

Esteja, pois, pronto para horas de lazer sem fim e, principalmente, para o aprendizado paulatino e ilustrado do BASIC Estendido usado nos "microps coloridos" da linha TRS-80.



ISBN 85-213-0292-4





सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख

सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख

सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख

सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख

सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख

सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख  
सुख

SOLUTIONS