Universidade Federal do Rio Grande do Sul
Instituto de Matemática
Cadernos de Matemática e Estatística
Série B: Trabalho de Apoio Dodático

A TARTARUGA NO ESPAÇO TRIDIMENSIONAL

Paulo Werlang de Oliveira
Elisabete Rambo
Suzana Lima dos Santos
Coordenação: Profa. Maria Alice Gravina

Série B, nº 9, FEV/92
Porto Alegre, fevereiro de 1992

SUMÁRIO:

I. INTRODUÇÃO:

O objetivo do nosso trabalho é desenvolver um software gráfico que permita ao usuário construir objetos no espaço tridimensional e então projetá-los sobre um plano de forma que possam ser representados , da melhor maneira possível , na tela do computador . Para tal , definiremos procedimentos que permitam :

a) comandar uma nova tartaruga "tridimensional" em suas trajetórias espaciais ;

b) a representação destas trajetórias em um plano .

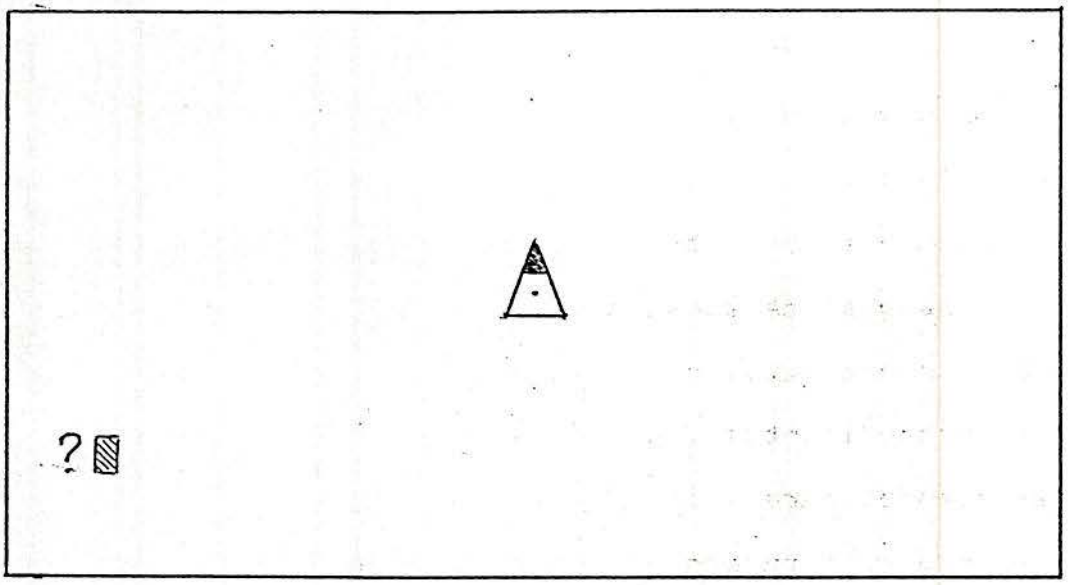
Estes procedimentos funcionarão de maneira semelhante aos comandos existentes para a tartaruga "plana" .

É importante dotarmos a tartaruga de movimentos no espaço tridimensional visto que este é um recurso poderoso , que pode ser usado para os mais diversos fins , como por exemplo :

- construir sólidos geométricos (como os poliedros) ;
- gerar superfícies de revolução (cones e paraboloides);
- descrever e representar a trajetória de partículas no espaço ;
- estudar a projeção de sombras .

II.A - A TARTARUGA NO PLANO:

Quando inicializamos o trabalho com a tartaruga no plano o que temos é :



a) a tartaruga no centro da tela

b) olhos apontando na direção vertical e braço direito apontando na direção horizontal.

Usando os comandos FD (forward) e BK (back) deslocamos a tartaruga para outro ponto do plano e através dos comandos RT (right) e LT (left) mudamos a direção dos olhos e braço.

Já implementado na linguagem LOGO temos um sistema de coordenadas cartesiano com origem no centro da tela e eixos X e Y respectivamente nas direções horizontal e vertical.

Com isto temos que num dado instante o estado da tartaruga fica determinado por :

a) um par de números reais (x,y) que corresponde as coordenadas do ponto no plano em que se encontra a tartaruga.

b) um par de vetores [v1,v2] onde v1 e v2 são vetores no plano, ortogonais , de comprimento unitário , correspondendo respectivamente a direção dos olhos e braço direito da tartaruga.

Vamos representar o estado da tartaruga por [P , B] onde P = (x,y) e B = [v1,v2].

II.B - A TARTARUGA NO ESPAÇO:

1. INTRODUÇÃO:

Na implementação da tartaruga tridimensional vamos pensar de modo análogo à situação no plano.

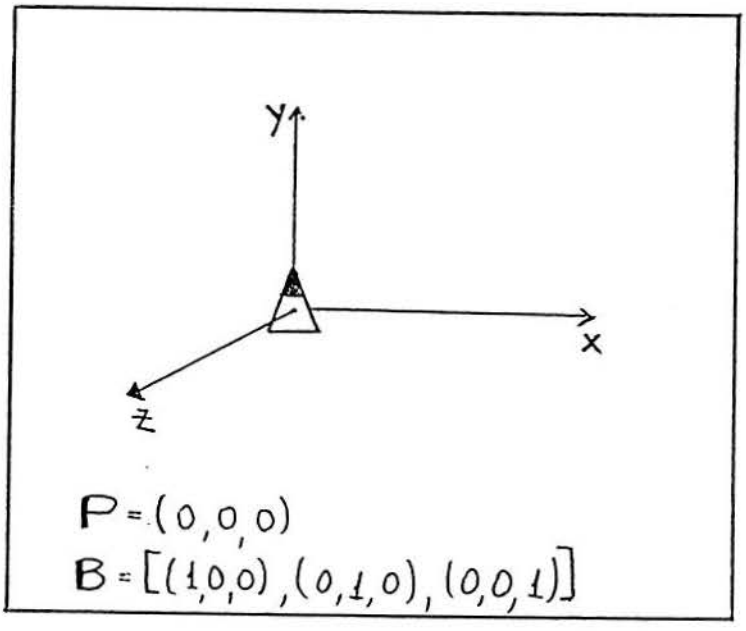
Começamos escolhendo um sistema referencial cartesiano XYZ.

O estado da tartaruga fica definido por :

a) Uma tripla de números reais (x,y,z) que corresponde as coordenadas do ponto do espaço em que se encontra a tartaruga.

b) Uma tripla de vetores [d,o,p] onde d, o, p são vetores no espaço , ortogonais e de comprimento unitário, correspondentes respectivamente a direção do braço direito, direção dos olhos e direção do casco da tartaruga.

Vamos representar o estado da tartaruga por [P , B] onde P = (x,y,z) e B = [d,o,p] e vamos nos referir a P como posição da tartaruga e a B como triedro-orientacão ou simplesmente orientação da tartaruga.



O estado da tartaruga pode ser modificado através de mudanças de posição e/ou de mudanças de orientação, com os procedimentos que veremos a seguir.

2. MOVIMENTOS DA TARTARUGA NO ESPAÇO:

2.1 Inicialização :

Para começarmos a trabalhar com a tartaruga tridimensional precisaremos antes de mais nada definir o seu estado inicial.

Vamos tomar P = (0,0,0) e B = [(1,0,0),(0,1,0),(0,0,1)]

Procedimento :

```
TO INICIATAT
MAKE "X 0 MAKE "Y 0 MAKE "Z 0
MAKE "OX 0 MAKE "OY 1 MAKE "OZ 0
MAKE "DX 1 MAKE "DY 0 MAKE "DZ 0
MAKE "PX 0 MAKE "PY 0 MAKE "PZ 1
END
```

2.2 Mudança de Orientação :

Definiremos a seguir procedimentos que mudam o triedro-orientação da tartaruga. Mudanças neste triedro serão obtidas através de três rotações básicas :

1) Rotação dos vetores olhos e braço direito, de um ângulo &, no plano por eles definido, movimento chamado VIRA .

2) Rotação dos vetores olhos e perpendicular , de um ângulo &, no plano por eles definido, movimento chamado CABECEA.

3) Rotação dos vetores braço direito e perpendicular, de um ângulo &, no plano por eles definido, movimento chamado BALANCEA.

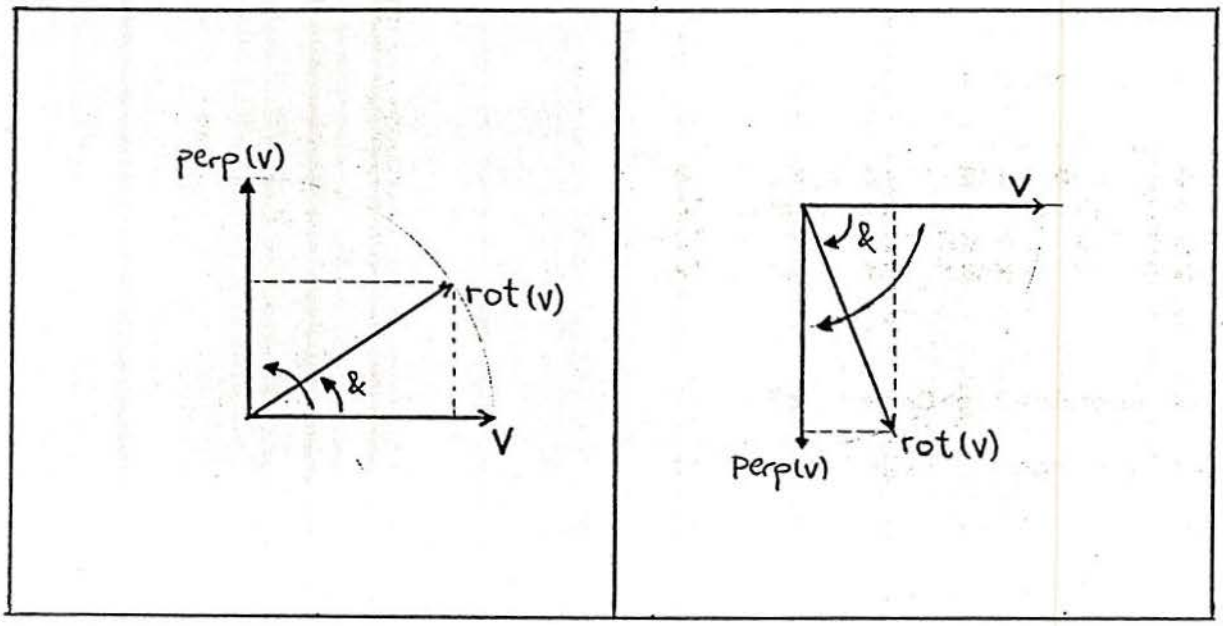
Para descrevermos estes movimentos precisamos entender como é feita a rotação de um vetor em um plano, em torno de um eixo que seja perpendicular a este plano.

Para isto sejam :

Π : plano em $\mathbb{R}^3$ passando pela origem

v : vetor no plano Π .

Girando v de um ângulo & no plano Π, obtemos um novo vetor, que vamos representar por rot(v). Girando v de um ângulo de 90 graus no plano Π, no mesmo sentido tomado acima, obtemos um vetor perpendicular que vamos representar por perp(v).



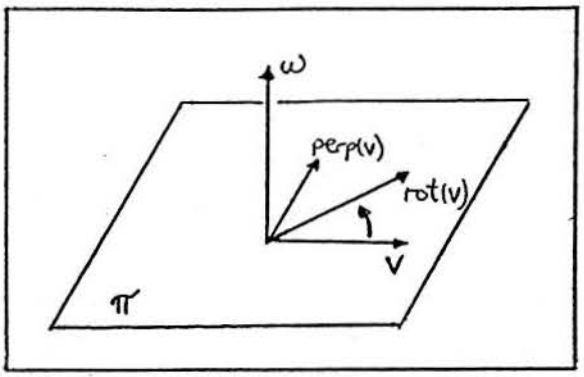
Queremos expressar rot(v) em termos de v e perp(v).

Do teorema de Pitágoras aplicado à qualquer uma das situações acima obtemos :

$$rot(v) = \cos(\&) * v + \operatorname{sen}(\&) * perp(v) \qquad (1)$$

Se w é um vetor perpendicular a π, podemos relativamente à w nos referir a rotação em π como horária ou anti-horária do seguinte modo :

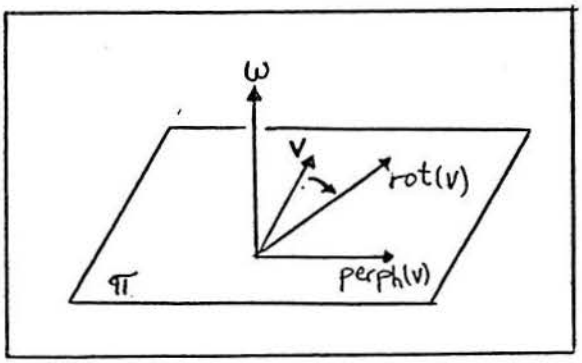
a) rotação anti-horária:



Significa que olhando para π a partir da extremidade de w, o giro de v para rot(v) se dá no sentido contrário aos ponteiros do relógio.

Neste caso vamos nos referir à perp(v) como vetor perpendicular a v no sentido anti-horário, e vamos denotá-lo por perpah(v).

b) rotação horária :



Significa que olhando para π a partir da extremidade de w, o giro de v para rot(v) se dá no sentido dos ponteiros do relógio. Neste caso vamos nos referir à perp(v) como vetor perpendicular a v no sentido horário, e vamos denotá-lo por perph(v).

Usando que perp$_h$(v) = - perp$_h$(v) , cos(&) = cos(-&) e sen(-&) = - sen(&) e mais a fórmula obtida em (1) temos que :

$$\boxed{rot(v) = \cos(\&) * v + sen(\&) * perp_h(v)} \quad (2)$$

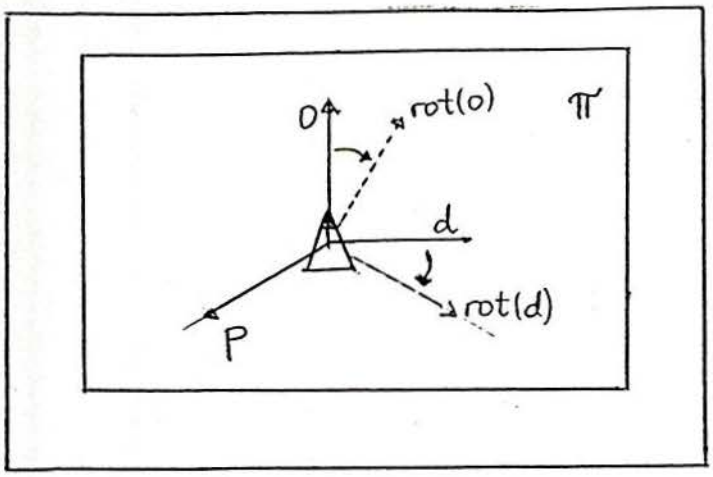
Sendo que quando: & > 0 o giro é horário ;

& < 0 o giro é anti-horário.

Ou seja, obtemos uma única expressão para rot(v), sendo que o sinal de & nos informa sobre o sentido de rotação.

Feito isto estamos prontos para descrever os movimentos da tartaruga.

VIRA



Tomando o vetor $w = p$ temos que :

perp(o) = d e perp(d) = -o, donde

rot(o) = cos(&) * o + sen(&) * d

rot(d) = cos(&) * d - sen(&) * o

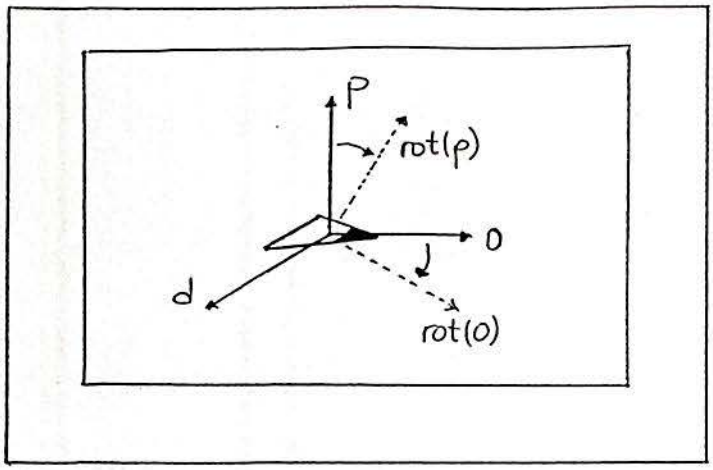
O PROCEDIMENTO VIRA

O procedimento VIRA tem como entrada o ângulo a ser girado:

```
TO VIRA :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :OX + :SE * :DX
MAKE "TY :CO * :OY + :SE * :DY
MAKE "TZ :CO * :OZ + :SE * :DZ
MAKE "DX :CO * :DX - :SE * :OX
MAKE "DY :CO * :DY - :SE * :OY
MAKE "DZ :CO * :DZ - :SE * :OZ
MAKE "OX :TX
MAKE "OY :TY
MAKE "OZ :TZ
END
```

Observamos que no procedimento acima (tx,ty,tz) são apenas variáveis auxiliares que servem para armazenar temporariamente os valores das coordenadas do novo vetor o .

CABECEA



Tomando w = d temos que :

perph(p) = o e perph(o) = -p , donde

rot(p) = cos(&) * p + sen(&) * o

rot(o) = cos(&) * o - sen(&) * p

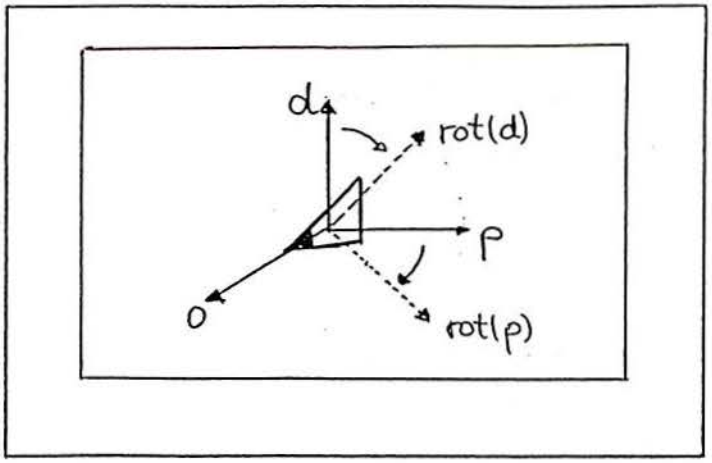
O PROCEDIMENTO CABECEA

O procedimento CABECEA tem como entrada o ângulo a ser girado:

```
TO CABECEA :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :OX - :SE * :PX
MAKE "TY :CO * :OY - :SE * :PY
MAKE "TZ :CO * :OZ - :SE * :PZ
MAKE "PX :CO * :PX + :SE * :OX
MAKE "PY :CO * :PY + :SE * :OY
MAKE "PZ :CO * :PZ + :SE * :OZ
MAKE "OX :TX
MAKE "OY :TY
MAKE "OZ :TZ
END
```

Observamos que no procedimento acima (tx,ty,tz) são apenas variáveis auxiliares que servem para armazenar temporariamente os valores das coordenadas do novo vetor o .

BALANCEA



Tomando $w = o$ temos que :

$perp_h(d) = p$ e $perp_h(p) = -d$, donde

$rot(p) = \cos(\&) * p - \sen(\&) * d$

$rot(d) = \cos(\&) * d + \sen(\&) * p$

O PROCEDIMENTO BALANCEA

O procedimento BALANCEA tem como entrada o ângulo a ser girado:

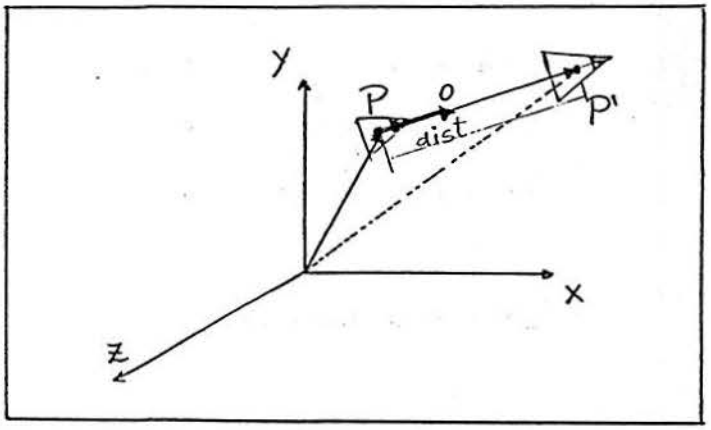
```
TO BALANCEA :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :DX + :SE * :PX
MAKE "TY :CO * :DY + :SE * :PY
MAKE "TZ :CO * :DZ + :SE * :PZ
MAKE "PX :CO * :PX - :SE * :DX
MAKE "PY :CO * :PY - :SE * :DY
MAKE "PZ :CO * :PZ - :SE * :DZ
MAKE "DX :TX
MAKE "DY :TY
MAKE "DZ :TZ
END
```

Observamos que no procedimento acima (tx,ty,tz) são apenas variáveis auxiliares que servem para armazenar temporariamente os valores das coordenadas do novo vetor d .

2.3 Mudança de Posição :

Situada a tartaruga em um ponto P = (x,y,z) do espaço, modificamos sua posição deslocando-a na direção do vetor olhos, conservando a sua orientação atual $\mathcal{B}$ = [d,o,p] .

Para isto precisamos entender como é feita a adição de vetores. Seja P = (x,y,z) posição da tartaruga e $\mathcal{B}$ = [d,o,p] triedro. Se P' é a nova posição , então , por soma de vetores obtemos : P' = P + dist * o ; onde dist é a distância que a tartaruga deve percorrer .



O PROCEDIMENTO ANDA

O procedimento ANDA tem como entrada a distância a ser avançada:

```
TO ANDA :DISTANCIA
MAKE "X :X + :DISTANCIA * :OX
MAKE "Y :Y + :DISTANCIA * :OY
MAKE "Z :Z + :DISTANCIA * :OZ
PROJECAO :X :Y :Z
END
```

Observação :

O procedimento "PROJECAO" que aparece no programa acima é assunto a ser tratado mais adiante . Este procedimento é que permite a representação de pontos do espaço na tela do computador; para isto o que precisamos é identificar o ponto P = (x,y,z) do espaço com um par de números reais . Esta identificação depende do tipo de projeção que vamos escolher : projeção em perspectiva ou projeção em paralelo (Capítulo IV) . Além do tipo de projeção devemos levar em conta também a posição do observador do movimento . Este é o assunto que veremos a seguir .

III. O OBSERVADOR:

1. INTRODUÇÃO :

Quando a tartaruga se desloca no espaço ela vai deixando uma trilha, um rastro, o qual chamaremos de trajetória real da tartaruga. Note que este rastro será visto de maneira diferente se o olharmos de posições diferentes. Como a idéia é representar a trajetória da tartaruga da melhor maneira possível na tela do computador, e esta representação depende tanto do ponto em que se encontra o observador e da maneira como ele está olhando, precisamos definir procedimentos que modifiquem o estado do observador tal qual fizemos com a tartaruga.

Da mesma forma que definimos o estado da tartaruga, definiremos agora o estado do observador por :

a) um triedro de vetores ortonormais semelhante ao da tartaruga, dito orientação do observador, indicando a direção em que o observador está olhando. Este triedro será representado por :

   do = (d1,d2,d3) vetor braço direito do observador
   oo = (o1,o2,o3) vetor olhos do observador
   po = (p1,p2,p3) vetor perpendicular do observador.

b) um número real, denotado por :dist , que dá a distância do observador ao plano determinado pelos vetores do , po, dito plano de projeção . Este plano será o plano de projeção usado para fazermos a representação dos movimentos da tartaruga.

Agora as possíveis mudanças de estado do observador são as seguintes : podemos mudar a direção para onde o observador está olhando, mudando apenas a orientação do seu triedro T = [do, oo, po] com procedimentos semelhantes aos vira , cabecea, balancea da tartaruga e podemos mudar também a distância que existe entre ele e o plano de projeção , trazendo-o para perto ou afastando-o deste plano.

2. MOVIMENTOS DO OBSERVADOR :

2.1 Inicialização :

Vamos considerar o estado inical do observador como sendo :

oo = (0,0,-1)
do = (1,0,0)
po = (0,1,0)

Ou seja , ao inicializarmos o triedro do observador ele está olhando para dentro da tela do computador , seu braço direito aponta na direção do eixo X e seu vetor perpendicular aponta para o eixo Y. O procedimento OROB permite definir os valores iniciais para o triedro orientação do observador.

```
TO OROB   (ORienta OBservador)
MAKE "O1 0 MAKE "O2 0 MAKE "O3 -1
MAKE "D1 1 MAKE "D2 0 MAKE "D3 0
MAKE "P1 0 MAKE "P2 1 MAKE "P3 0
END
```

O procedimento POSOB permite ao usuário escolher e modificar a distância do observador ao plano de projeção . A distância inicial é 500 e é definida dentro do programa INICIAOBS a seguir.

```
TO POSOB :NDDD   (POSiciona OBservador)
MAKE "DIST :NDDD
END
```

O programa INICIAOBS é que na verdade inicializa o observador ao começarmos a trabalhar com a tartaruga tridimensional.

```
TO INICIAOBS   (INICIAliza OBSservador)
OROB
POSOB 500
END
```

2.2 Mudanças na Orientação do Observador :

Os programas VOB , COB , BOB respectivamente viram , cabeceam e balanceam o observador . São definidos de maneira análoga aos procedimentos vira , cabecea e balancea da tartaruga.

```
TO VOB :ANGULO    (Vira OBservador)
MAKE "CO COS -:ANGULO
MAKE "SE SIN -:ANGULO
MAKE "TX :CO*:O1+:SE*:D1
MAKE "TY :CO*:O2+:SE*:D2
MAKE "TZ :CO*:O3+:SE*:D3
MAKE "D1 :CO*:D1-:SE*:O1
MAKE "D2 :CO*:D2-:SE*:O2
MAKE "D3 :CO*:D3-:SE*:O3
MAKE "O1 :TX
MAKE "O2 :TY
MAKE "O3 :TZ
END
```

```
TO COB :ANGULO    (Cabecea OBservador)
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO*:O1-:SE*:P1
MAKE "TY :CO*:O2-:SE*:P2
MAKE "TZ :CO*:O3-:SE*:P3
MAKE "P1 :CO*:P1+:SE*:O1
MAKE "P2 :CO*:P2+:SE*:O2
MAKE "P3 :CO*:P3+:SE*:O3
MAKE "O1 :TX
MAKE "O2 :TY
MAKE "O3 :TZ
END
```

```
TO BOB :ANGULO    (Balancea OBservador)
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO*:D1+:SE*:P1
MAKE "TY :CO*:D2+:SE*:P2
MAKE "TZ :CO*:D3+:SE*:P3
MAKE "P1 :CO*:P1-:SE*:D1
MAKE "P2 :CO*:P2-:SE*:D2
MAKE "P3 :CO*:P3-:SE*:D3
MAKE "D1 :TX
MAKE "D2 :TY
MAKE "D3 :TZ
END
```

2.3 Mudança na Distância do Observador à Origem :

é feita mediante o procedimento POSOB acima.

O procedimento POSOB permite ao usuário escolher e modificar a distância do observador ao plano de projeção . A distância inicial é 500 e é definida dentro do programa INICIAOBS .

```
TO POSOB :NDDD   (POSiciona OBservador)
MAKE "DIST :NDDD
END
```

Observação :

Como a posição do observador é definida por P = -dist * oo convém ressaltar que os procedimentos VIRAOBSERVADOR (VOB) e CABECEAOBSERVADOR (COB) acima , além de mudar a orientação do observador também modificam a sua posição .

IV. A REPRESENTAÇÃO NA TELA DO COMPUTADOR :

1. INTRODUÇÃO:

Para que possamos representar um objeto tridimensional num plano qualquer precisamos escolher algum tipo de projeção. As projeções mais utilizadas são a projeção em paralelo e a projeção em perspectiva. A projeção em paralelo matematicamente é mais simples , mas tem a desvantagem de representar os objetos sempe do mesmo tamanho , independente da distância que está sendo observado (fig 1). Já a projeção em perspectiva , matematicamente não tão simples , tem a vantagem de ser mais realista , já que neste caso o tamanho da representação muda conforme a distância de observação (fig 2).

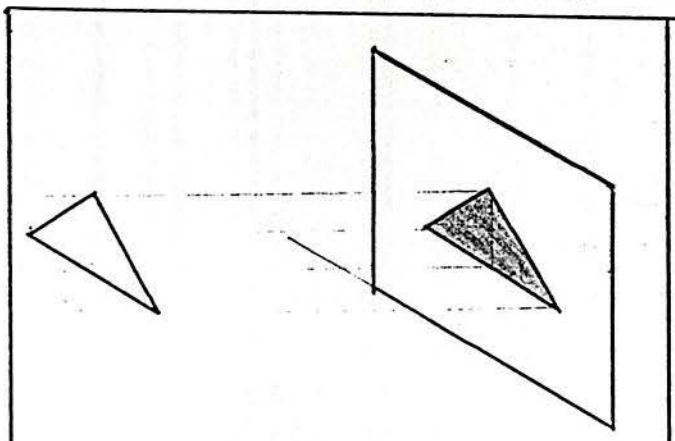


fig.1 (a)

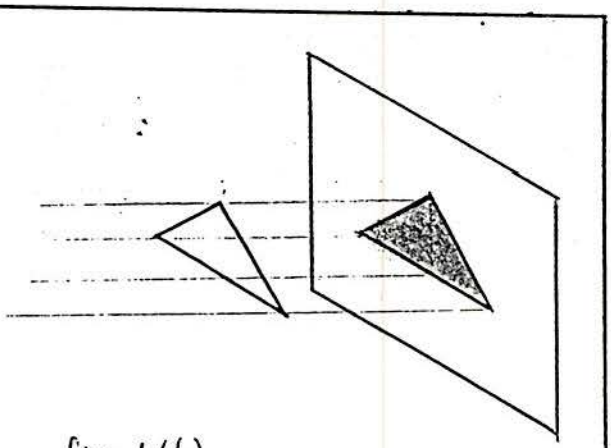


fig.1 (b)

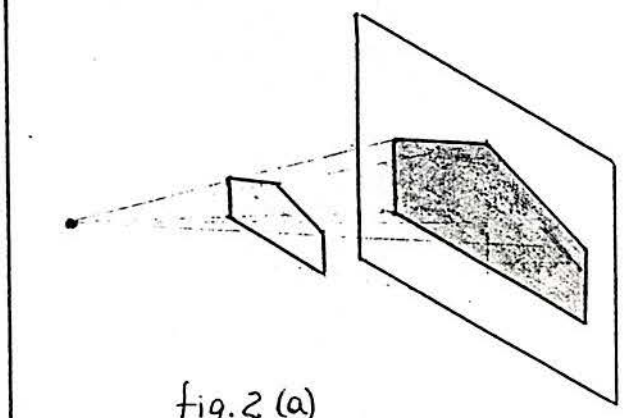


fig.2 (a)

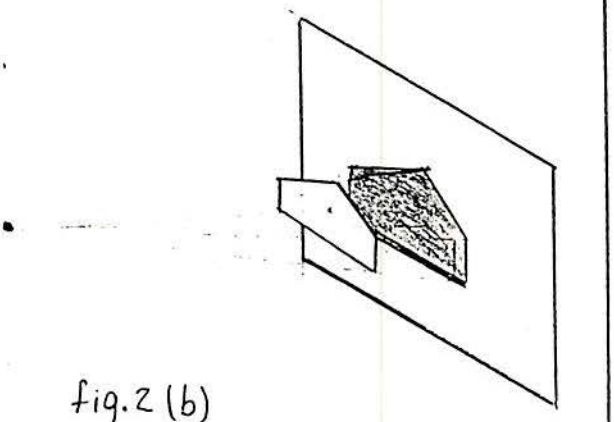


fig.2 (b)

2. PROJEÇÕES EM PARALELO E EM PERSPECTIVA:

Para obter as expressões das projeções precisamos preliminarmente de alguns resultados de matemática. Assim :

Seja $\mathbb{R}^3 = \{ (x,y,z) / x, y, z \in \mathbb{R} \}$. Sempre que necessário, identificaremos $P = (x,y,z)$ com a seta com origem em $O=(0,0,0)$ e extremidade em (x,y,z) e vamos nos referir aos elementos de $\mathbb{R}^3$ como pontos ou vetores.

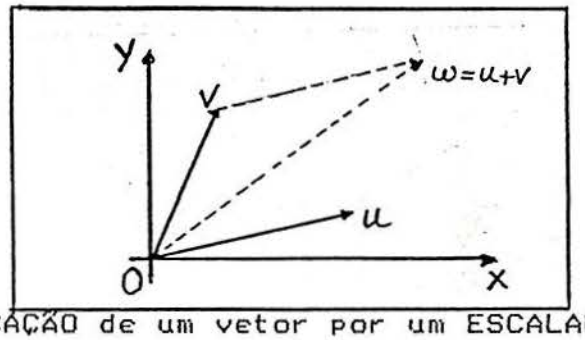
Em $\mathbb{R}^3$ vamos considerar as operações :

a) SOMA de vetores $u , v \in \mathbb{R}^3$.

Algebricamente temos :

$w = u+v = (u1+v1, u2+v2, u3+v3)$ onde $u = (u1,u2,u3)$ e $v = (v1,v2,v3)$.

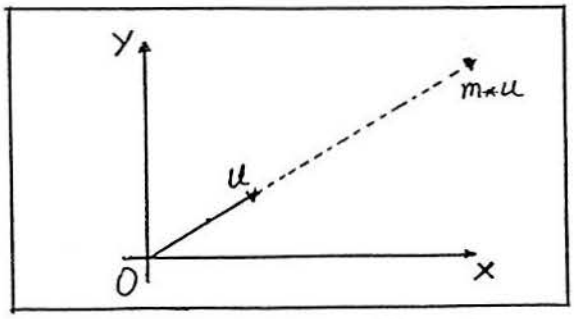
Geometricamente temos, pela regra dos paralelogramos:



b) MULTIPLICAÇÃO de um vetor por um ESCALAR :

Algebricamente temos :

$w = m * u = (m * u1 , m * u2 , m * u3)$ onde $m \in \mathbb{R}$ e $u=(u1,u2,u3)$



Queremos em $\mathbb{R}^3$ trabalhar com a noção de comprimento de vetor e ortogonalidade entre vetores. Portanto consideraremos o produto escalar .

-- PRODUTO ESCALAR :

.Definição :

Dados dois vetores u , $v \in \mathbb{R}^3$ tais que $u = (x1,y1,z1)$ e $v = (x2,y2,z2)$ definimos o produto escalar de u por v como:

$$u.v = x1*x2 + y1*y2 + z1*z2$$

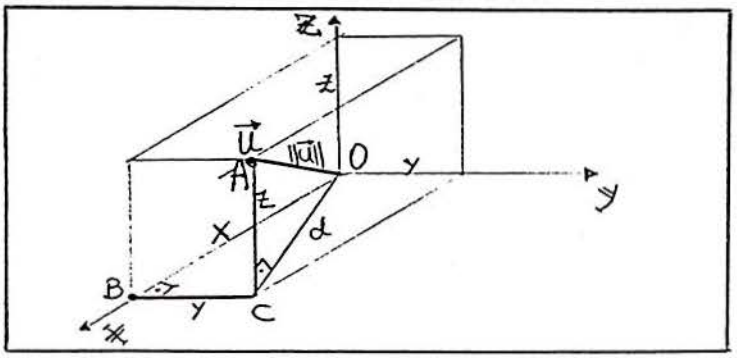
.Propriedades do produto escalar :

1) $u.v = v.u$
2) $u.(v+w)=u.v+u.w$
3) $(m*u).v=m*(u.v)$ onde m e um numero real qualquer.
4) $u.u \geq 0$ com $u.u = 0 \iff u = (0,0,0)$.

Dado $u \in \mathbb{R}^3$ definimos a NORMA ou MODULO de u como :

$$\| u \| = \sqrt{u.u}$$

Geometricamente a norma de u corresponde ao comprimento euclidiano da seta à u identificada. De fato :



$\| u \| = OA$ Olhando no triângulo retângulo OBC sai ,
$X = OB$ por Pitágoras, que $(OC)^2 = (OB)^2 + (BC)^2$.
$Y = BC$ Logo $d^2 = X^2 + Y^2$. Portanto $(OC)^2 = X^2 + Y^2$.
$Z = CA$ Olhando no triângulo retângulo OCA sai ,
$d = OC$ também por Pitágoras, que :
$\| u \|^2 = (OA)^2 = (OC)^2 + (CA)^2 = X^2 + Y^2 + Z^2$.

Logo , conclui-se que $\| u \| = \sqrt{X^2 + Y^2 + Z^2}$ é o comprimento euclidiano do vetor u .

Teorema : Dados dois vetores u , $v \in \mathbb{R}^3$ temos que :

$$\boxed{u.v = \| u \| . \| v \| . \cos(\&)}$$

onde & é o ângulo entre os dois vetores .

Ora sendo $u = v + w$ segue que $w = u - v$. Daí , pela lei dos cossenos vem que :

$\| w \|^2 = \| u \|^2 + \| v \|^2 - 2 . \| u \| . \| v \| . \cos(\&)$

mas :

$\| w \|^2 = w.w = (u-v).(u-v) = u.u - u.v - v.u + v.v =$

$= u.u + v.v - 2\, u.v$

$\| u \|^2 = u.u$

$\| v \|^2 = v.v$

daí :

$u.u + v.v - 2\, u.v = u.u + v.v - 2 . \| u \| . \| v \| . \cos(\&)$

logo : $u.v = \| u \| . \| v \| . \cos(\&)$

Segue da igualdade acima que u e v são ortogonais (isto é , o ângulo entre eles mede 90°) se e somente se $u.v = 0$.

- BASE ORTONORMAL:

Se $u, v, w \in \mathbb{R}^3$ são vetores tais que $\| u \| = \| v \| = \| w \| = 1$ e $u \perp v$, $u \perp w$, $v \perp w$ então dizemos que $\mathcal{B} = (u, v, w)$ é uma base ortonormal de $\mathbb{R}^3$. Numa base ortonormal $\mathcal{B} = (u, v, w)$ temos que as coordenadas do vetor r nesta base são :

$$\boxed{[r] = (r.u \ , \ r.v \ , \ r.w)}$$

De fato :

- TEOREMA DA MUDANÇA DE BASES :

Se $\mathbb{B} = (u,v,w)$ é base ortonormal de $\mathbb{R}^3$ e $r \in \mathbb{R}^3$ é um vetor então as coordenadas de r na base $\mathbb{B}$ são $[r.u, r.v, r.w]$, isto é

$$r = (r.u)*u + (r.v)*v + (r.w)*w \ .$$

Notação : $[r] = (r.u , r.v , r.w)$

Prova :

Como $\mathbb{B}$ é base de $\mathbb{R}^3$ então existem $a, b, c \in \mathbb{R}$ tais que

$$r = a * u + b * v + c * w$$

Daí segue que $r.u = (a * u + b * v + c * w).u =$

$$= a * u.u + b * v.u + c * w.u =$$

$$= a * \|u\|^2 + b * 0 + c * 0 =$$

$$= a * 1 = a \ .$$

Logo $a = r.u$

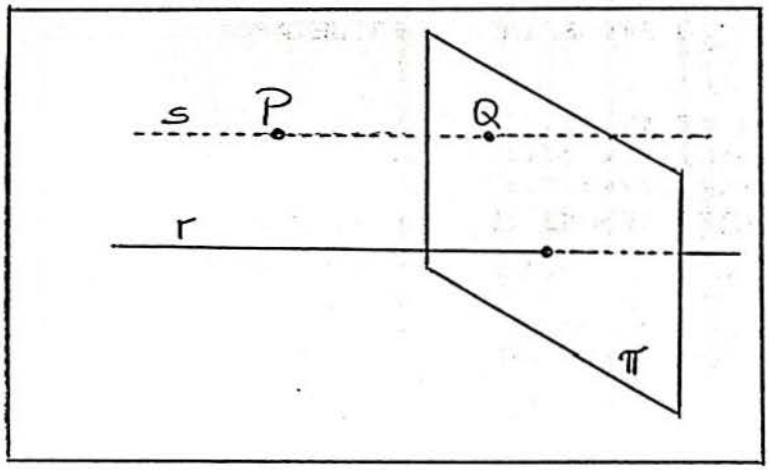
Da mesma forma sai que $b = r.v$ e que $c = r.w$.

A PROJEÇÃO EM PARALELO :

A noção geométrica da projeção em paralelo :

Fixa-se um plano π de projeção e uma reta r que intercepta π em um único ponto . Dado um ponto P qualquer, fazemos a projeção em paralelo do ponto P da seguinte maneira :

Traçamos a reta s paralela à reta r pelo ponto P e encontramos o ponto Q de intersecção de s com π . Por definição este ponto Q é a projeção em paralelo do ponto P no plano π e com direção dada pela reta r .



Como é livre a escolha da reta r e do plano π , escolheremos r e π de forma a simplificar ao máximo os cálculos envolvidos . Desta forma vamos adotar como plano de projeção o plano formado pelos vetores **do** e **po** do triedro do observador . A distância do observador ao plano de projeção será medida na direção do vetor **-oo** . Resta apenas escolher a reta que dará a direção da projeção . Faremos isto escolhendo como direção da projeção a direção dada pelo vetor olhos do observador . Note que a base (do , po , oo) é ortonormal e portanto o feixe de retas paralelas é sempre perpendicular ao plano de projeção .

Com esta construção , a expressão analítica da projecão em paralelo é muito símples . Basta desconsiderarmos a terceira coordenada do ponto P escrito na base $\mathbb{B}$ = {do,po,oo}

Assim a projeção em paralelo do ponto P sobre o plano determinado por (do,po) tem como coordenadas Q = (P.do,P.po) que será o ponto que vamos representar na tela do microcomputador usando o sistema de coordenadas cartesiano já implementado pela linguagem LOGO .

O PROCEDIMENTO PROJETAPAR

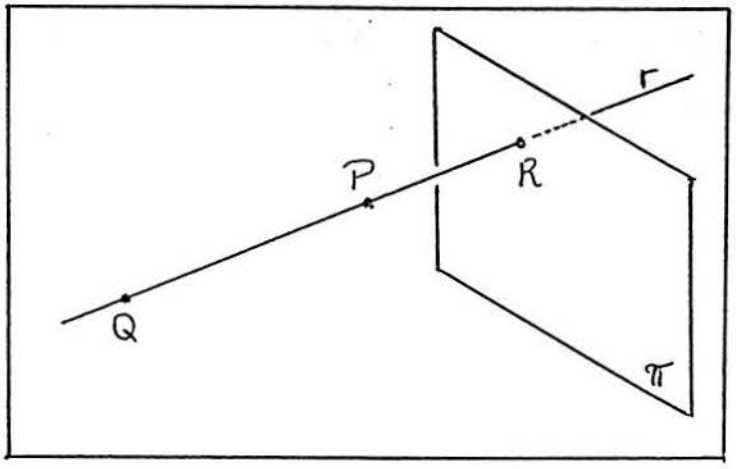
```
TO PROJETAPAR :X :Y :Z
MAKE "E1 (:X*:D1+:Y*:D2+:Z*:D3)
MAKE "E2 (:X*:P1+:Y*:P2+:Z*:P3)
MAKE "LISTA LIST (ROUND :E1)(ROUND :E2)
SETPOS :LISTA
END
```

A PROJEÇÃO EM PERSPECTIVA :

Noção geométrica da projeção em perspectiva :

Para fazermos uma projeção em perspectiva precisamos apenas fixar um plano de projeção π e um ponto Q fora de π que servirá como suporte para a projeção . Este ponto Q geralmente é chamado de ponto de vista da projeção . A projeção em perspectiva é feita da seguinte maneira :

Dado um ponto P qualquer , traçamos a reta r que passa por P e por Q e encontramos o ponto R de intersecção desta reta r com o plano π . Se existir , esse ponto R será o projetado de P em π . Da mesma forma que fizemos na projeção em paralelo , vamos fixar como plano de projeção o plano dado por (do , po) e o ponto de vista será colocado justamente na posição do observador POSOB , ou seja , em -dist*oo .



Queremos a seguir determinar as coordenadas do ponto R na base $\mathcal{B}$ = (do , po) . Para isto seja :

π = (a*du + b*po / a, b $\in \mathbb{R}$) = [a,b,0] : plano de projeção

P = (x,y,z) = [c1,c2,c3] com c1 = P.do , c2 = P.po e c3 = P.oo

Como POSOB = -dist * ao temos POSOB = [0,0,-dist]

Temos u = P - POSOB = [c1 , c2 , c3 + dist] e a reta que passa por POSOB e P é dada por :

r = { POSOB + m*u / m∈ℝ } = {[m*c1,m*c2,m*(c3+dist)-dist] / m∈ℝ}

Sendo t a intersecção da reta r com o plano π temos :

t = POSOB + m0*u = [m0*c1,m0*c2,m0*(c3+dist)-dist] = [**,##,0]

Daí t ∈ π ⟺ m0*(c3+dist)-dist = 0

⟺ m0*(c3+dist) = dist

⟺ m0 = dist/(dist+c3) .

Daí temos :

t = [(dist/(dist+c3)) * c1 , (dist/(dist+c3)) * c2 , 0] =

= (dist/(dist+c3)) * [c1 , c2 , 0] .

Logo t = (dist/(dist+c3)) * [c1 , c2] é o ponto que precisa ser plotado na tela do computador .

O PROCEDIMENTO PROJETAPERP

```
TO PROJETAPERP :X :Y :Z
MAKE "C1 (:X*:D1+:Y*:D2+:Z*:D3)
MAKE "C2 (:X*:P1+:Y*:P2+:Z*:P3)
MAKE "C3 (:X*:O1+:Y*:O2+:Z*:O3)
MAKE "FF (:DIST/(:DIST+:C3))
MAKE "LISTA LIST (ROUND (:FF*:C1))(ROUND (:FF*:C2))
SETPOS :LISTA
END
```

COMPACTANDO PROCEDIMENTOS :

Voce certamente notou a semelhança entre os procedimentos PROJETAPAR e PROJETAPERP . A idéia agora é unir os dois procedimentos e fazer um único que sirva tanto para fazer projeções em paralelo como em perspectiva , de modo que você possa escolher entre um tipo ou outro à vontade .

Se você olhar bem , vai notar que o ponto t a ser marcado na tela gráfica na projeção em perspectiva é o mesmo ponto da projeção em paralelo corrigido por um fator de amplificação , isto é , t = FF * Q onde FF = dist/(dist+c3) é o fator de amplificação .

Note que se o fator de amplificação FF for tal que FF = 1 então t = Q é a projeção em paralelo , caso contrário a projeção é em perspectiva . Aí , se usarmos o procedimento PROJETAPERP e criarmos um meio do computador decidir se queremos que a projeção seja em paralelo ou em perspectiva o trabalho está pronto . Faremos isto criando uma variável que armazene o tipo de projeção que queremos e então mandamos o computador atualizar o valor do fator de amplificação FF conforme o caso . O nome desta variável será OPCAOPROJECAO e se a opção de projeção for "paralela" o computador fará o fator de amplificação FF valer 1 senão FF valerá dist/(dist+c3) .

Como fica o novo procedimento :

```
TO PROJECAO :X :Y :Z
MAKE "C1 (:X*:D1+:Y*:D2+:Z*:D3)
MAKE "C2 (:X*:P1+:Y*:P2+:Z*:P3)
MAKE "C3 (:X*:O1+:Y*:O2+:Z*:O3)
IF :OPCAOPROJECAO = "PARALELA [MAKE "FF 1][MAKE "FF (:DIST/(:DIST+:C3))]
MAKE "LISTA LIST (ROUND (:FF*:C1))(ROUND (:FF*:C2))
SETPOS :LISTA
END
```

Resta agora fazermos um procedimento que permita ao usuário escolher entre uma ou outra projeção . O procedimento abaixo faz isso :

```
TO OPCAODEPROJECAO
CT
PR [Se voce quiser projecao em Paralelo digite : < P > ]
PR []
PR [Se voce quiser projecao em peRspectiva digite : < R > ]
PR []
MAKE "OPPROJ RC
IF :OPPROJ = "P [MAKE "OPCAOPROJECAO "PARALELA][MAKE "OPCAO
PROJECAO "PERSPECTIVA]
CT
END
```

Não esqueça de chamar este programa pelo menos uma vez no início para que a variável OPCAOPROJECAO tenha seu valor armazenado . Não esqueça também de mudar o nome do novo procedimento PROJECAO nas demais rotinas .

APÊNDICE A

GUIA DO USUÁRIO :

Este guia foi feito com a intenção de servir como um manual de referência e consulta rápida dos comandos da tartaruga tridimensional. Para que você possa aproveitá-lo da melhor maneira possível daremos a seguir um exemplo de como consultá-lo:

```
==================================================================
FORWARD (1) [FD] (2)                                        (3) (I)
------------------------------------------------------------------
Desloca a Tartaruga plana . (4)
------------------------------------------------------------------
(5) * SINTAXE  :   FORWARD  :distancia

(6) * EXEMPLO  :   FORWARD 10

(7) * DESCRIÇÃO:   FORWARD  desloca  a  tartaruga ,  mantendo
       inalterada a sua orientação. Se a entrada for positiva a
       tartaruga  anda  para  frente, caso  contrário  vai  para
       trás. A entrada pode ser tanto um número , uma  variável
       ou uma expressão que resulte num valor numérico.

(8) * PROGRAMA EXEMPLO :      TO MOVIMENTOALEATORIO
                              MAKE "DISTANCIA 10*RANDOM 6
                              MAKE "ANGULO RANDOM 361
                              FORWARD :DISTANCIA RIGHT :ANGULO
                              MOVIMENTOALEATORIO
                              END
==================================================================
```

(1) : Neste campo vai o nome completo da primitiva. Este nome precisa ser digitado em letras maiúsculas e sem espaços em branco.

(2) : Neste campo vai a abreviatura da primitiva , se existir.

(3) : Indicação da caráter da primitiva. Dá a característica da primitiva, segundo o código abaixo :

- (C) - Comando de modo direto (não requer argumentos de entrada).
- (I) - Instrução (sempre precisa de algum argumento de entrada).
- (F) - Indica que a primitiva é alguma função matemática , necessitando de algum outro comando atuando junto , como por exemplo os comandos de atribuição MAKE ou impressão PRINT .

(4) : Descrição resumida da primitiva .

(5) : Modelo de sintaxe correto, com respectivas entradas.

(6) : Exemplo .

(7) : Explicação detalhada do funcionamento da primitiva, com ressalvas à seu uso.

(8) : Programa exemplo . O programa exemplo deve sempre ser executado a fim de elucidar bem o funcionamento da primitiva.

ANDA [A] (I)

Desloca a tartaruga tridimensional segundo uma direção.

* SINTAXE : ANDA :distancia

* EXEMPLO : ANDA 25

* DESCRIÇÃO: A instrução ANDA serve para deslocar a tartaruga tridimensional, sem afetar a sua orientação. A distancia de deslocamento é dada pela entrada e a direção é determinada pelo vetor olhos da tartaruga. O sentido de deslocamento é para frente se a entrada for positiva e para trás se negativa. O argumento pode ser tanto uma constante, uma variável ou uma expressão cujo resultado seja um valor numérico.

* PROGRAMA EXEMPLO :

```
TO ANDANDO :DELTAX
ANDA :DELTAX  V 90
ANDA -2*:DELTAX  V 90
ANDANDO (:DELTAX+10)
END
```

BALANCEA [B] (I)

Balancea a tartaruga por um determinado ângulo .

* SINTAXE : BALANCEA :angulo

* EXEMPLO : BALANCEA 30

* DESCRIÇÃO: A instrução BALANCEA faz mudar a orientação da tartaruga , mudando o seu triedro orientação , através de uma rotação cujo eixo de rotação é o vetor olhos o , não afetando a posição da tartaruga. Se a entrada for positiva a rotação será no sentido horário para quem olha da extremidade do vetor olhos o e anti-horário se a entrada for negativa. O argumento de entrada pode ser tanto uma constante , uma variável ou uma expressão numérica , que após calculada será reduzida ao intervalo [0°,360°]

BALANCEAOBSERVADOR [BOB] (I)

Balancea o observador por um determinado ângulo .

* SINTAXE : BALANCEAOBSERVADOR :angulo

* EXEMPLO : BALANCEAOBSERVADOR 10

* DESCRIÇÃO: A instrução BALANCEAOBSERVADOR muda a orientação do observador , rotando o seu triedro orientação , por um determinado ângulo dado como entrada. A instrução BOB não muda a posição do observador. O eixo de rotação é determinado pelo vetor olhos oo do observador. Se a entrada for positiva a rotação será no sentido horário para quem olha da extremidade do vetor olhos oo e anti-horário se a entrada for negativa. O argumento de entrada pode ser tanto uma constante , uma variável ou uma expressão numérica , que após calculada será reduzida ao intervalo [0°,360°].

CABECEA [C] (I)

Cabecea a tartaruga por um determinado ângulo .

* SINTAXE : CABECEA :angulo

* EXEMPLO : CABECEA 25

* DESCRIÇÃO: A instrução CABECEA faz mudar a orientação da tartaruga , mudando o seu triedro orientação , através de uma rotação cujo eixo de rotação é o vetor direita d , não afetando a posição da tartaruga. Se a entrada for positiva a rotação será no sentido horário para quem olha da extremidade do vetor direita d e anti-horário se a entrada for negativa. O argumento de entrada pode ser tanto uma constante , uma variável ou uma expressão numérica , que após calculada será reduzida ao intervalo [0°,360°].

CABECEAOBSERVADOR [COB] (I)

Cabecea o observador por um determinado ângulo .

* SINTAXE : CABECEAOBSERVADOR :angulo

* EXEMPLO : CABECEAOBSERVADOR -45

* DESCRIÇÃO: A instrução CABECEAOBSERVADOR faz mudar a orientação do observador , mudando o seu triedro orientação , através de uma rotação cujo eixo é o vetor direita do observador do . Esta rotação afeta a posição do observador , uma vez que ela é determinada por -:dist*oo . Se a entrada for positiva a rotação será no sentido horário para quem olha da extremidade do vetor direita do e anti-horário se a entrada for negativa. O argumento de entrada pode ser tanto uma constante , uma variável ou uma expressão numérica , que após calculada será reduzida ao intervalo [0°,360°].

H [] (C)

Orienta a tartaruga .

* SINTAXE : H

* EXEMPLO : H

* DESCRIÇÃO: O comando H serve para reorientar a tartaruga, colocando-a com a orientação do estado inicial, que é : o=(0,1,0) ; d=(1,0,0) e p=(0,0,1). A posição da tartaruga permanece inalterada.

I [] (C)

Inicializa o sistema da tartaruga tridimensional .

* SINTAXE : I

* EXEMPLO : I

* DESCRIÇÃO: Este comando inicializa as variáveis do sistema , limpa a tela gráfica e a tela texto . Coloca a tartaruga e o observador em suas posições e orientações de partida.

INICIAOBS [] (C)

Inicializa o observador .

* SINTAXE : INICIAOBS

* EXEMPLO : INICIAOBS

* DESCRIÇÃO: O comando INICIAOBS inicializa o estado do observador, colocando-o no ponto P=(0,0,500) e com orientação dada pelos vetores do=(1,0,0) ; po=(0,1,0) e oo=(0,0,-1). A distância ao plano de projeção é fixada em 500.

INICIATAT [] (C)

Inicializa a tartaruga .

* SINTAXE : INICIATAT

* EXEMPLO : INICIATAT

* DESCRIÇÃO: O comando INICIATAT serve para inicializar o estado da tartaruga tridimensional, colocando-a na origem do sistema de coordenadas (Posição O=(0,0,0)) e com triedro-orientação dado pelos vetores d=(1,0,0) ; o=(0,1,0) e p=(0,0,1). Desta forma o estado inicial da tartaruga tridimensional coincide com o estado de partida da tartaruga plana , ou seja , no centro da tela e apontando para o norte .

OPCAODEPROJECAO [] (C)

Permite escolher o tipo de projeção desejada .

* SINTAXE : OPCAODEPROJECAO

* EXEMPLO : OPCAODEPROJECAO

* DESCRIÇÃO: O comando OPCAODEPROJECAO serve para que se possa escolher o tipo de projeção adequada ao tipo de problema que se está resolvendo. As opções apresentadas são projeção em paralelo e projeção em perspectiva. Ao ser executado , este comando fica a espera que o usuário digite uma das seguintes teclas : <P> para a projeção em paralelo ou <R> para projeção em perspectiva.

ORIENTAOBSERVADOR [OROB] (C)

Orienta o Observador .

* SINTAXE : ORIENTAOBSERVADOR

* EXEMPLO : OROB

* DESCRIÇÃO: O comando OROB serve para reorientarmos o observador, colocando-o no seu triedro orientação de partida, que é do=(1,0,0) ; po=(0,1,0) e oo=(0,0,-1). Note que este comando também muda automaticamente a posição do observador , pois ela fica determinada pelo vetor v=(-dist)*oo . O que permanece inalterado com este comando é a distância do observador ao plano de projeção.

P [] (I)

Posiciona a tartaruga num determinado ponto do espaço .

* SINTAXE : P :coordenadax :coordenaday :coordenadaz

* EXEMPLO : P 10 20 30

* DESCRIÇÃO: A instrução P serve para colocar a tartaruga num determinado ponto do espaço tridimensional. Os argumentos de entrada são as três coordenadas do ponto em que se quer situar a tartaruga. Estes argumentos podem ser constantes , variáveis ou expressões cujo resultado forneça um valor numérico real.

POSOB [] (I)

Posiciona o observador a uma determinada distância do plano de projeção .

* SINTAXE : POSOB :distancia

* EXEMPLO : POSOB 500

* DESCRIÇÃO: A instrução POSOB permite afastar ou aproximar o observador do plano de projeção, não afetando o triedro orientação do observador. Quanto maior o valor do argumento mais longe o observador estará do plano de projeção e vice-versa. O argumento de entrada pode ser tanto uma constante , uma variável ou uma expressão numérica cujo valor dê como resultado um número positivo.

PROJECAO [] ()

Projeta um ponto no plano .

* SINTAXE : PROJECAO :cx :cy :cz

* EXEMPLO : PROJECAO 10 20 30

* DESCRIÇÃO: PROJECAO é um procedimento que serve para podermos representar um determinado ponto do espaço na tela do computador . Ele é usado internamente, dentro dos demais procedimentos que fazem a mudança de posição da tartaruga. Você pode usá-lo sozinho , mas precisa ter muito cuidado porque daí os pontos que serão projetados não tem nenhuma relação com a tartaruga tridimensional.

U.V [U.V] (F)

Calcula o produto "interno" ou escalar de dois vetores .

* SINTAXE : U.V :ux :uy :uz :vx :vy :vz

* EXEMPLO : U.V 1 0 1 3 4 5

* DESCRIÇÃO: U.V é uma função especial que calcula o produto escalar de dois vetores dados. Seus argumentos de entrada são as 3 coordenadas do vetor u seguidas das 3 coordenadas do vetor v , nesta ordem . No exemplo acima , foi feito o produto escalar do vetor u=(1,0,1) pelo vetor v=(3,4,5). u e v podem ser quaisquer vetores do espaço tridimensional. Esta função é usada internamente no procedimento projeção e serve apenas para torná-lo mais eficiente , mas você pode usá-la sempre que quizer fazer o produto escalar de dois vetores. Por exemplo, se você quizer saber se os vetores (-1,2,0) e (2,1,-2) são perpendiculares , basta você digitar a seguinte linha de instruções : PR U.V -1 2 0 2 1 -2 e ver se o resultado é zero ou não.

VIRA [V] (I)

Vira a tartaruga por um determinado ângulo .

* SINTAXE : VIRA :angulo

* EXEMPLO : VIRA -30

* DESCRIÇÃO: A instrução VIRA faz mudar a orientação da tartaruga , mudando o seu triedro orientação , através de uma rotação cujo eixo é o vetor perpendicular p, não afetando a posição da tartaruga. Se a entrada for positiva a rotação será no sentido horário para quem olha da extremidade do vetor perpendicular p e anti-horário se a entrada for negativa. O argumento de entrada pode ser tanto uma constante , uma variável ou uma expressão numérica , que após calculada será reduzida ao intervalo [0°,360°] .

VIRAOBSERVADOR [VOB] (I)

Vira o observador por um determinado ângulo .

* SINTAXE : VIRAOBSERVADOR :angulo

* EXEMPLO : VIRAOBSERVADOR -50

* DESCRIÇÃO: A instrução VIRAOBSERVADOR faz mudar a orientação do observador , mudando o seu triedro orientação , através de uma rotação cujo eixo é o vetor perpendicular do observador po . Esta rotação afeta a posição do observador , uma vez que ela é determinada por -:dist*oo . Se a entrada for positiva a rotação será no sentido anti-horário para quem olha da extremidade do vetor perpendicular po e horário se a entrada for negativa. O argumento de entrada pode ser tanto uma constante , uma variável ou uma expressão numérica , que após calculada será reduzida ao intervalo [0°,360°].

APÊNDICE B

LISTA DE PALAVRAS RESERVADAS DA TARTARUGA TRIDIMENSIONAL

A

ANDA

B

BALANCEA

BALANCEAOBSERVADOR

BOB

C

CABECEA

CABECEAOBSERVADOR

COB

H

I

INICIAOBS

INICIATAT

OPCAODEPROJECAO

ORIENTAOBSERVADOR

OROB

P

POSOB

PROJECAO

U.V

V

VIRA

VIRAOBSERVADOR

VOB

TARTARUGA TRIDIMENSIONAL - PROCEDIMENTOS :

A

```
TO A :DISTANCIA
MAKE "X :X + :DISTANCIA * :OX
MAKE "Y :Y + :DISTANCIA * :OY
MAKE "Z :Z + :DISTANCIA * :OZ
PROJECAO :X :Y :Z
END
```

ANDA

```
TO ANDA :DISTANCIA
MAKE "X :X + :DISTANCIA * :OX
MAKE "Y :Y + :DISTANCIA * :OY
MAKE "Z :Z + :DISTANCIA * :OZ
PROJECAO :X :Y :Z
END
```

B

```
TO B :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :DX + :SE * :PX
MAKE "TY :CO * :DY + :SE * :PY
MAKE "TZ :CO * :DZ + :SE * :PZ
MAKE "PX :CO * :PX - :SE * :DX
MAKE "PY :CO * :PY - :SE * :DY
MAKE "PZ :CO * :PZ - :SE * :DZ
MAKE "DX :TX
MAKE "DY :TY
MAKE "DZ :TZ
END
```

BALANCEA

```
TO BALANCEA :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :DX + :SE * :PX
MAKE "TY :CO * :DY + :SE * :PY
MAKE "TZ :CO * :DZ + :SE * :PZ
MAKE "PX :CO * :PX - :SE * :DX
MAKE "PY :CO * :PY - :SE * :DY
MAKE "PZ :CO * :PZ - :SE * :DZ
MAKE "DX :TX
MAKE "DY :TY
MAKE "DZ :TZ
END
```

BALANCEAOBSERVADOR

```
TO BALANCEAOBSERVADOR :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :D1 + :SE * :P1
MAKE "TY :CO * :D2 + :SE * :P2
MAKE "TZ :CO * :D3 + :SE * :P3
MAKE "P1 :CO * :P1 - :SE * :D1
MAKE "P2 :CO * :P2 - :SE * :D2
MAKE "P3 :CO * :P3 - :SE * :D3
MAKE "D1 :TX
MAKE "D2 :TY
MAKE "D3 :TZ
END
```

BOB

```
TO BOB :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :D1 + :SE * :P1
MAKE "TY :CO * :D2 + :SE * :P2
MAKE "TZ :CO * :D3 + :SE * :P3
MAKE "P1 :CO * :P1 - :SE * :D1
MAKE "P2 :CO * :P2 - :SE * :D2
MAKE "P3 :CO * :P3 - :SE * :D3
MAKE "D1 :TX
MAKE "D2 :TY
MAKE "D3 :TZ
END
```

C

```
TO C :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :OX - :SE * :PX
MAKE "TY :CO * :OY - :SE * :PY
MAKE "TZ :CO * :OZ - :SE * :PZ
MAKE "PX :CO * :PX + :SE * :OX
MAKE "PY :CO * :PY + :SE * :OY
MAKE "PZ :CO * :PZ + :SE * :OZ
MAKE "OX :TX
MAKE "OY :TY
MAKE "OZ :TZ
END
```

CABECEA

```
TO CABECEA :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :OX - :SE * :PX
MAKE "TY :CO * :OY - :SE * :PY
MAKE "TZ :CO * :OZ - :SE * :PZ
MAKE "PX :CO * :PX + :SE * :OX
MAKE "PY :CO * :PY + :SE * :OY
MAKE "PZ :CO * :PZ + :SE * :OZ
MAKE "OX :TX
MAKE "OY :TY
MAKE "OZ :TZ
END
```

CABECEAOBSERVADOR

```
TO CABECEAOBSERVADOR :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :O1 - :SE * :P1
MAKE "TY :CO * :O2 - :SE * :P2
MAKE "TZ :CO * :O3 - :SE * :P3
MAKE "P1 :CO * :P1 + :SE * :O1
MAKE "P2 :CO * :P2 + :SE * :O2
MAKE "P3 :CO * :P3 + :SE * :O3
MAKE "O1 :TX
MAKE "O2 :TY
MAKE "O3 :TZ
END
```

COB

```
TO COB :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :O1 - :SE * :P1
MAKE "TY :CO * :O2 - :SE * :P2
MAKE "TZ :CO * :O3 - :SE * :P3
MAKE "P1 :CO * :P1 + :SE * :O1
MAKE "P2 :CO * :P2 + :SE * :O2
MAKE "P3 :CO * :P3 + :SE * :O3
MAKE "O1 :TX
MAKE "O2 :TY
MAKE "O3 :TZ
END
```

H

```
TO H
MAKE "OX 0 MAKE "OY 1 MAKE "OZ 0
MAKE "DX 1 MAKE "DY 0 MAKE "DZ 0
MAKE "PX 0 MAKE "PY 0 MAKE "PZ 1
END
```

I

```
TO I
CS CT INICIATAT
RISCA
MAKE "ULTP "DENTRO
INICIAOBS
END
```

INICIAOBS

```
TO INICIAOBS
MAKE "DIST 500
MAKE "O1 0 MAKE "O2 0 MAKE "O3 -1
MAKE "D1 1 MAKE "D2 0 MAKE "D3 0
MAKE "P1 0 MAKE "P2 1 MAKE "P3 0
END
```

INICIATAT

```
TO INICIATAT
MAKE "LISTA [0 0]
PU SETPOS :LISTA PD
MAKE "ULTP "DENTRO
MAKE "X 0 MAKE "Y 0 MAKE "Z 0
MAKE "OX 0 MAKE "OY 1 MAKE "OZ 0
MAKE "DX 1 MAKE "DY 0 MAKE "DZ 0
MAKE "PX 0 MAKE "PY 0 MAKE "PZ 1
END
```

OPCAODEPROJECAO

```
TO OPCAODEPROJECAO
CT
PR []
PR [Se voce quizer projecao :]
PR []
PR [em Paralelo _______> tecle <P>]
PR [em peRspectiva ___> tecle <R>]
PR [] PR []
MAKE "OPPROJ RC
IF :OPPROJ = "P [MAKE "OPCAOPROJECAO "PARALELA] [MAKE "OPCAOPROJECAO "PERS
PECTIVA]
END
```

ORIENTAOBSERVADOR

```
TO ORIENTAOBSERVADOR
MAKE "O1 0 MAKE "O2 0 MAKE "O3 -1
MAKE "D1 1 MAKE "D2 0 MAKE "D3 0
MAKE "P1 0 MAKE "P2 1 MAKE "P3 0
END
```

OROB

```
TO OROB
MAKE "O1 0 MAKE "O2 0 MAKE "O3 -1
MAKE "D1 1 MAKE "D2 0 MAKE "D3 0
MAKE "P1 0 MAKE "P2 1 MAKE "P3 0
END
```

P

```
TO P :XX :YY :ZZ
MAKE "X :XX
MAKE "Y :YY
MAKE "Z :ZZ
PROJECAO :X :Y :Z
END
```

POSOB

```
TO POSOB :NDDD
MAKE "DIST :NDDD
END
```

PROJECAO

```
TO PROJECAO :X :Y :Z
MAKE "E1 U.V :X :Y :Z :D1 :D2 :D3
MAKE "E2 U.V :X :Y :Z :P1 :P2 :P3
MAKE "E3 U.V :X :Y :Z :O1 :O2 :O3
IF :OPCAOPROJECAO = "PARALELA [MAKE "FF 1] [MAKE "FF (:DIST / (:DIST + :E3
))]
MAKE "LISTA LIST (ROUND (:FF * :E1)) (ROUND (:FF * :E2))
IF (:OPCAOPROJECAO = "PARALELA) [ESCONDESNPAR (FIRST :LISTA) (LAST :LISTA)
] [ESCONDESNPERP (FIRST :LISTA) (LAST :LISTA)]
SETPOS :LISTA
END
```

U.V

```
TO U.V :X1 :Y1 :Z1 :X2 :Y2 :Z2
OUTPUT (:X1 * :X2 + :Y1 * :Y2 + :Z1 * :Z2)
END
```

V

```
TO V :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :OX + :SE * :DX
MAKE "TY :CO * :OY + :SE * :DY
MAKE "TZ :CO * :OZ + :SE * :DZ
MAKE "DX :CO * :DX - :SE * :OX
MAKE "DY :CO * :DY - :SE * :OY
MAKE "DZ :CO * :DZ - :SE * :OZ
MAKE "OX :TX
MAKE "OY :TY
MAKE "OZ :TZ
END
```

VIRA

```
TO VIRA :ANGULO
MAKE "CO COS :ANGULO
MAKE "SE SIN :ANGULO
MAKE "TX :CO * :OX + :SE * :DX
MAKE "TY :CO * :OY + :SE * :DY
MAKE "TZ :CO * :OZ + :SE * :DZ
MAKE "DX :CO * :DX - :SE * :OX
MAKE "DY :CO * :DY - :SE * :OY
MAKE "DZ :CO * :DZ - :SE * :OZ
MAKE "OX :TX
MAKE "OY :TY
MAKE "OZ :TZ
END
```

VIRAOBSERVADOR

```
TO VIRAOBSERVADOR :ANGULO
MAKE "CO COS -:ANGULO
MAKE "SE SIN -:ANGULO
MAKE "TX :CO * :O1 + :SE * :D1
MAKE "TY :CO * :O2 + :SE * :D2
MAKE "TZ :CO * :O3 + :SE * :D3
MAKE "D1 :CO * :D1 - :SE * :O1
MAKE "D2 :CO * :D2 - :SE * :O2
MAKE "D3 :CO * :D3 - :SE * :O3
MAKE "O1 :TX
MAKE "O2 :TY
MAKE "O3 :TZ
END
```

VOB

```
TO VOB :ANGULO
MAKE "CO COS -:ANGULO
MAKE "SE SIN -:ANGULO
MAKE "TX :CO * :O1 + :SE * :D1
MAKE "TY :CO * :O2 + :SE * :D2
MAKE "TZ :CO * :O3 + :SE * :D3
MAKE "D1 :CO * :D1 - :SE * :O1
MAKE "D2 :CO * :D2 - :SE * :O2
MAKE "D3 :CO * :D3 - :SE * :O3
MAKE "O1 :TX
MAKE "O2 :TY
MAKE "O3 :TZ
END
```

TOTAL DE PROCEDIMENTOS :25

APÊNDICE C

Como a listagem dos procedimentos anexada foi escrita para a versão LOGO 1.0 (IBM-PC) , damos a seguir um dicionário de conversão de comandos que permitirá ao usuário da versão LOGO 1.0 para computadores padrão MSX re-escrever facilmente tais procedimentos na linguagem do seu micro .

DICIONARIO DE PRIMITIVAS :

LOGO : Versao LCSI 1.0 (IBM-PC)	LOGO : Versao HOT-LOGO 1.0 (MSX)
- .CALL	- .chame
- .CONTENTS	- primitivas
- .DEPOSIT	- .deposite
- .EXAMINE	- .examine
- .SCRUNCH	- proporcao
- .SETSCRUNCH	- mudeproporcao
- AND	- e
- ARCTAN	- arctan
- ASCII	- ascii
- BACK	- paratras
- BF	- sp
- BK	- pt
- BL	- su
- BUTFIRST	- semprimeiro
- BUTLAST	- semultimo
- BUTTONP	- ebotao
- CHAR	- caractere
- CLEAN	- apaguedesenho
- CLEARSCREEN	- paracentroa
- CLEARTEXT	- apaguetexto
- COPYDEF	- copie

- COS ..... - cos
- COUNT ..... - num.elem
- CS ..... - pca
- CT ..... - att
- CURSOR ..... - cursor
- DEFINE ..... - defina
- DEFINEDP ..... - eprocedimento
- DIFFERENCE ..... - diferenca
- DIR ..... - arquivos
- DOT ..... - ponhaponto
- DRIBBLE"LPT1 ..... - comimpressora
- ED ..... - ed
- EDIT ..... - edite
- EDNS ..... - edns
- EMPTYP ..... - evazia
- END ..... - fim
- EQUALP ..... - saoiguais
- ER ..... - el
- ERALL ..... - eltudo
- ERASE ..... - elimine
- ERASEFILE ..... - eliminearq
- ERN ..... - eln
- ERNS ..... - elns
- ERPS ..... - elps
- FALSE ..... - falso
- FD ..... - pf
- FENCE ..... - aumentelimite
- FILL ..... - pinte
- FIRST ..... - primeiro
- FORWARD ..... - parafrente

FPUT	juntenoinicio
HEADING	direcao
HIDETURTLE	desaparecatat
HOME	paracentro
HT	dt
IF	se
INT	int
ITEM	elemento
KEYP	temcar
LAST	ultimo
LEFT	paraesquerda
LIST	lista
LISTP	elista
LOAD	carregue
LOADPIC	carreguedes
LPUT	juntenofim
LT	pe
MAKE	atribua
NAME	coloque
NODES	memlivre
NODRIBBLE	semimpressora
NOT	nao
NUMBERP	enumero
OP	envie
OR	algum
OUTPUT	envie
PAUSE	nivelinicial
PD	ul
PE	ub
PENDOWN	uselapis

- PENERASE ..... - useborracha
- PENREVERSE ..... - useinversor
- PENUP ..... - usenada
- PO ..... - mop
- POALL ..... - motudo
- POFILE ..... - mostrearq
- PONS ..... - mons
- POPS ..... - mops
- POS ..... - posicao
- POTS ..... - mots
- PPS ..... - moprop
- PR ..... - esc
- PRIMITIVEP ..... - eprimitiva
- PRINT ..... - escreva
- PRODUCT ..... - produto
- PU ..... - un
- PX ..... - ui
- QUOTIENT ..... - quociente
- RANDOM ..... - sorteieate
- RC ..... - care
- READCHAR ..... - car.entrada
- READLIST ..... - lin.entrada
- RECYCLE ..... - liberemem
- REMAINDER ..... - resto
- REPEAT ..... - repita
- RERANDOM ..... - reproduza
- RIGHT ..... - paradireita
- RL ..... - line
- ROUND ..... - arredonde
- RT ..... - pd

- RUN	.....	- faca
- SAVE	.....	- gravetudo
- SAVEPIC	.....	- gravedes
- SE	.....	- sn
- SENTENCE	.....	- sentenca
- SETBG	.....	- mudecf
- SETCURSOR	.....	- mudecursor
- SETH	.....	- mudedc
- SETHEADING	.....	- mudedc
- SETPEN	.....	- mudecl
- SETPOS	.....	- mudepos
- SETSHAPE	.....	- mudefig
- SETTC	.....	- mudecor
- SETTEXT	.....	- mudeteto
- SETX	.....	- mudex
- SETY	.....	- mudey
- SHAPE	.....	- figura
- SHOW	.....	- mostra
- SHOWNP	.....	- evisivel
- SHOWTURTLE	.....	- aparecatat
- SIN	.....	- sen
- SQRT	.....	- raizq
- ST	.....	- at
- STAMP	.....	- carimbe
- STOP	.....	- pare
- SUM	.....	- soma
- TEXT	.....	- texto
- THING	.....	- conteudo
- TO	.....	- aprenda
- TONE	.....	- toque

- TOWARDS ..... - direcaopara
- TRUE ..... - verd
- TYPE ..... - ponha
- WAIT ..... - espere
- WORD ..... - palavra
- WORDP ..... - epalavra
- WRAP ..... - tirelimite
- XCOR ..... - coorx
- YCOR ..... - coory

TOTAL DE PRIMITIVAS :150

BIBLIOGRAFIA :

ABELSON,H.,DISESSA,A. Turtle Geometry: the computer as a medium for exploring Mathematics. Cambridge: MIT Press, 1980.

ROSELLÓ,L.R. LOGO: de la tortuga a la inteligencia artificial. Vector Ediciones, 1986.

PAPERT,S. LOGO: computadores e educação. São Paulo: Brasiliense, 1985.

SUMÁRIO:

BIBLIOGRAFIA :

ABELSON,H.,DISESSA,A. Turtle Geometry: the computer as a medium for exploring Mathematics. Cambridge: MIT Press, 1980.

ROSELLó,L.R. LOGO: de la tortuga a la inteligencia artificial. Vector Ediciones, 1986.

PAPERT,S. **LOGO:** computadores e educação. São Paulo: Brasiliense, 1985.

Publicações do Instituto de Matemática da UFRGS
Cadernos de Matemática e Estatística

Série B: Trabalho de Apoio Didático

1. Elsa Mundstock - Curso Básico Sobre Wordstar 3.45 - MAR/89.
2. Jaime B. Ripoll - Introdução ao Cálculo Diferencial Via Funções de Uma Variável Real - OUT/89.
3. Edmund R. Puczylowski - Dimension of Modular Lattices - JUN/90
4. Marcos Sebastiani - Geometrias Não Euclidianas - JUL/90
5. Sandra R. C. Pizzato - Cálculo Numérico - AGO/91
6. Vera Clotilde G. Carneiro - Elementos de Cálculo para Biologia - AGO/91
7. Elsa Mundstock - Iniciaç ao ao SPSS/PC - SET/91
8. Elisa Haag, Loiva C. de Zeni, Maria Alice Gravina e Vera Clotilde - Notas da 1a. Oficina de Matemática da UFRGS - JAN/92
9. Paulo Werlang de Oliveira, Elisabete Rambo, Suzana Lima dos Santos, Coordenaç ao: Profa. Maria Alice Gravina - A Tartaruga no Espaço Tridimensional - FEV/92

Universidade Federal do Rio Grande Sul
Reitor: Professor Tuiskon Dick

Instituto de Matemática
Diretor: Professor Aron Taitelbaum
Núcleo de Atividades Extra Curriculares
Coordenador: Professor Jaime Bruck Ripoll
Secretária: Faraildes Beatriz da Silva

Os Cadernos de Matemática e Estatística publicam as seguintes séries:
Série A: Trabalho de Pesquisa
Série B: Trabalho de Apoio Didático
Série C: Colóquio de Matemática SBM/UFRGS
Série D: Trabalho de Graduação
Série E: Dissertações de Mestrado
Série F: Trabalho de Divulgação
Série G: Textos para Discussão

Toda correspondência com solicitação de números publicados e demais informações deverá ser enviada para:

NAEC - Núcleo de Atividades Extra Curriculares
Instituto de Matemática - UFRGS
Av. Bento Gonçalves, 9500
91.500 - Agronomia - POA/RS
Telefone: 36.98.22 ou 39.13.55 Ramal: 6176