Ce manuel n'est pas une introduction. Il suppose que vous êtes déjà au courant de Logo ou d'un langage du même genre.

Si c'est la première fois que vous rencontrez Logo, vous devriez étudier d'abord le volume Introduction à la programmation par le graphique Tortue, qui accompagne le système Apple Logo. Les programmeurs expérimentés et les informaticiens pourront en apprendre suffisamment sur Logo en lisant directement le présent manuel.

Ce Manuel de référence donne des descriptions concises de chacune des primitives du langage Logo, ainsi que de nombreux échantillons de procédures (programmes). Les titres de chapitre donnés dans la table des matières serviront de référence commode quant à l'organisation de ces primitives.

Les définitions de procédures et les échantillons de dialogue entre l'usager et Logo sont imprimés en caractères différents du reste du manuel. Le but est de vous donner une représentation plus exacte de ce qui apparaît sur l'écran de texte. Dans certains cas, il y a une distinction supplémentaire, une couleur représentant les messages de l'ordinateur et l'autre représentant vos propres entrées. Nous avons pris une liberté supplémentaire: votre écran Apple présente jusqu'à 40 caractères par ligne; dans ce guide nous ne pouvons placer que 37 caractères par ligne.

On peut utiliser ce manuel de diverses manières. Si vous désirez savoir quelle est l'action d'une primitive particulière, vous pouvez la chercher dans l'index, ou consulter l'appendice A ou B pour référence rapide. Si vous cherchez une primitive pour exécuter une tâche particulière, vous pouvez consulter les titres de chapitres, qui indiquent les catégories de primitives. Vous pouvez également chercher des références dans l'index.

Exemple d'entrée
Voici une entrée de procédure typique:

DONNE

DONNE nom objet (commande)
DONNE veut deux arguments: le premier est un nom Logo, le deuxième est un objet Logo quelconque. DONNE est une commande (par opposition à une opération). Notez que le nom de la procédure est imprimé en majuscules. Les arguments, s'il en est, sont imprimés en italiques. Pour plus de commodité, on indique également le nom de

la procédure dans la marge de gauche.

Une entrée de procédure est suivie d'une description de son action, puis de quelques exemples d'utilisation. Les exemples sont donnés en caractères différents, pour mieux représenter ce qui apparaît sur l'écran de l'Apple.

Voici une entrée un peu plus complexe: .

ECRIS
EC

ECRIS objet (commande) abréviation: EC
(ECRIS objet1 objet2...) L'abréviation, EC, est un synonyme plus court de la commande ECRIS. D'ordinaire, ECRIS veut un argument, qui est un objet Logo quelconque. Mais on peut également donner à ECRIS un nombre quelconque d'arguments en même temps, pourvu que l'on enferme l'expression toute entière entre parenthèses. Cette variante est indiquée en dessous de la forme ordinaire.

Programmes Logo

Un programme Logo est une collection de procédures.

Par exemple, un programme destiné à tracer une maison se compose des procédures suivantes: MAISON, BOITE, TRI, DROITE, AVANCE, REPETE. Parmi celles-ci, les trois dernières sont des primitives. Les trois premières sont des procédures définies par l'usager, à partir de primitives Logo.

```
POUR MAISON
BOITE
AVANCE 50
DROITE 30
TRI 50
FIN

POUR BOITE
REPETE 4 [AVANCE 50 DROITE 90]
FIN

POUR TRI :TAILLE
REPETE 3 [AVANCE :TAILLE DROITE 120]
FIN
```

Nous supposons ici que vous avez pratiqué Logo quelque peu et que vous avez acquis une notion intuitive de ce qu'est Logo. Nous vous donnons ici un modèle plus formalisé. Ce modèle est une autre manière de se représenter Logo. Il n'est pas destiné à remplacer vos modes de pensée actuels, mais plutôt à les enrichir.

Logo formalisé

Le système Logo effectivement mis en oeuvre pour votre ordinateur Apple est très complexe. Il sera utile d'examiner une description formelle de Logo, avant de passer à sa réalisation effective. Nous commençons notre description formelle en nous restreignant à une partie de Logo que nous appellerons Logo formel; nous relâcherons ensuite nos restrictions pour décrire le langage en usage réel, que nous appellerons Logo relaxé.

Toutes les instructions du Logo formel sont valables sans modifications dans le Logo relaxé. Inversement, tout ce que l'on peut réaliser en Logo peut être réalisé dans le Logo formel; toutefois, vous allez immédiatement reconnaître des situations où le Logo formel contraint à des expressions que personne ne voudrait utiliser en pratique.

Par exemple, si vous avez l'habitude de Logo, vous remarquerez que les nombres ont une allure bizarre en Logo formel: ils sont cités. Comme les nombres sont des mots, il est possible de les citer dans le Logo relaxé également. Mais ceci se fait très rarement; dans le Logo relaxé, les nombres ont la propriété "d'autocitation".

Procédures et arguments: Nous supposons que vous avez une idée, au moins intuitive, de ce qu'est une procédure. Ci-dessous, nous allons élaborer des manières plus précises d'en parler.

Dans le Logo formel, chaque procédure exige un nombre défini d'arguments. Il y a deux sortes d'argument possibles: les mots et les listes. On peut les donner directement, comme dans l'expression AVANCE "100, ou indirectement par l'intermédiaire d'une autre procédure, par exemple dans l'expression AVANCE SOMME "60 "40. Ces deux expressions ont le même effet. Si un argument est donné directement sous forme d'un mot, il doit être précédé d'un guillemet, comme dans l'expression ECRIS "AVANCE. Si on le donne directement sous forme d'une liste, il doit être entouré de crochets carrés, comme dans l'expression ECRIS [COMMENT ALLEZ VOUS?].

Mots et listes: Un mot est composé de caractères. Une liste est composée d'éléments enfermés entre crochets carrés, séparés par des espaces; un élément est soit un mot (sans guillemets), soit une liste.

Expressions: Un nom de procédure suivi du nombre voulu d'arguments est une expression. Plus généralement, une expression est:

. un mot cité
. une liste, ou
. un nom de procédure (sans guillemets) suivi du nombre d'expressions voulu par la procédure.

Si ceci semble compliqué, considérons un exemple.
REPETE est une procédure qui exige deux arguments.

Voici une expression:
REPETE "3 [AV "10]

Ceci est également une expression:
REPETE SOMME "2 "1 [AV "10]

Dans ce cas le premier argument n'est pas un mot cité, mais une expression contenant une autre procédure, SOMME.

Voici une autre expression encore:
REPETE SOMME "3 "1 PHRASE "AV "10

Commandes et opérations: Logo comporte deux types de procédures. Les procédures qui (comme SOMME) produisent un objet Logo sont appelées opérations. Les autres (comme ECRIS) sont appelées commandes.

Ces définitions nous permettent de définir une instruction Logo.

Une instruction Logo est une expression d'un type particulier. Elle commence par un nom de procédure, qui doit être une commande. Toutes les autres procédures de l'expression doivent être des opérations.

Pourquoi est-il important de savoir si une expression est une instruction ou pas? Considerez

SOMME "3 "4

C'est une expression, mais non une instruction Logo.

SOMME va produire un nombre; mais si l'on écrit simplement SOMME "3 "4, on n'a pas dit quoi faire avec ce nombre. En fait, Logo retournera un message d'erreur, JE NE SAIS QUE FAIRE AVEC 7. Certains langages de programmation acceptent ces expressions, et écrivent tout simplement le 7. Dans la conception de Logo, nous avons préféré expliciter chaque action importante; si vous voulez que Logo écrive quelquechose, il faut le dire.

Logo relaxé
Tout ce qu'on peut faire en Logo peut se faire en Logo formel. Mais dans la mise en oeuvre du langage, nous avons permis certaines autres possibilités, soit pour rendre le langage plus naturel, soit pour minimiser la dactylographie.

Nous allons indiquer certaines relaxations importantes de Logo. D'autres sont mentionnées dans le corps de l'ouvrage. Voir également Appendice 1.

Nombres: En Logo formel, tous les mots utilisés comme arguments directs, y compris les nombres, doivent être cités. Dans le Logo relaxé, les nombres n'ont pas besoin de la marque de citation. Par exemple,

SOMME 3 4

Un nombre est un mot constitué de chiffres; il peut également contenir un signe moins, un point, et une lettre E ou N. Voir chapitre 4 (Opérations arithmétiques).

Procédures sous forme infixe: Logo formel n'utilise que des procédures sous forme préfixe. Ainsi, l'addition s'exprime par SOMME "3 "4. En Logo relaxé, on peut égale-

ment utiliser la forme infixe 3 + 4. De même pour la multiplication, la soustraction et la division. Voir chapitre 4 (Opérations arithmétiques).

Nombres d'arguments variables: En Logo formel chaque procédure a un nombre fixe d'arguments. En Logo relaxé, il y a plusieurs procédures qui peuvent accepter un nombre variable d'arguments. Si vous utilisez un nombre d'arguments différents du nombre normal, il faut enfermer l'expression entre parenthèses:

(SOMME 3 4 5 6 7 8)

Ces procédures n'exigent pas de parenthèses si le nombre d'arguments est inférieur au nombre normal, pourvu qu'il n'y ait rien d'autre à la suite sur la même ligne.

Deux points: Une caractéristique courante du Logo relaxé est l'utilisation des deuxpoints (:). En Logo formel, :X s'écrit CHOSE "X.

Parenthèses de groupement: Logo comporte des parenthèses qui permettent de déclarer explicitement à Logo la manière de grouper les éléments de l'expression. Par exemple considérez.

25 + 20 / 5

Faut-il effectuer l'addition d'abord, ce qui donnerait 9, ou la division d'abord, ce qui donnerait 29?

Le Logo relaxé suit la hiérarchie mathématique ordinaire, dans laquelle la multiplication et la division précèdent l'addition et la soustraction; ainsi l'opération indiquée commence par diviser 20 par 5, ce qui donne en fin de compte 29. Pour grouper les nombres de manière à exécuter l'addition en premier, on utilise des parenthèses:

(25 + 20) / 5

Commandes et opérations: En Logo formel, une procédure donnée est soit une commande soit une opération, mais jamais les deux. En Logo relaxé, il existe des procédures qui peuvent être utilisées parfois comme commandes et parfois comme opérations. EXECUTE et SI en sont des exemples. Voir chapitre 7 (Contrôle d'exécution).

La commande POUR: En Logo formel, on définie une procédure au moyen de la commande DEFINIS; cette commande prend deux arguments, un mot (nom de la procédure) et une liste (la définition). Toutes les règles habituelles concernant les guillemets et les crochets sont applicables sans exception:

```
DEFINIS "CARRE [[][REPETE 4[AV 100 DR 90]]]
```

En Logo relaxé, on peut définir une procédure au moyen d'EDITE et POUR. La commande POUR a deux aspects inhabituels. Elle assure automatiquement la citation de ses arguments, et elle vous permet de taper les lignes de la procédure sous forme interactive:

```
?POUR CARRE
>REPETE 4 [AV 100 DR 90]
>FIN
```

Note supplémentaire sur les opérations
En parlant de Logo formel, nous avons utilisé le mot produire: une procédure est dite produire un objet (appelé la sortie de la procédure). Dans la terminologie Logo ordinaire, nous disons que la procédure retourne un objet.

Les objets Logo
Il y a deux types de choses ou objets Logo: les mots et les listes.

Un mot, comme TÉLÉVISION ou 617, est composé de caractères (lettres, chiffres, ponctuation). Un nombre est un mot d'un type particulier.

Une liste, comme [LAPINS TÉLÉVISION 7[OREILLES PIED]], est composé d'objets Logo, dont chacun peut être un mot ou une liste. Voir chapitre 2 (Mots et listes).

Comment envisager les guillemets
Le rôle des guillemets se comprend bien dans l'exemple suivant:

```
?ECRIS "CAP
CAP
?ECRIS CAP
180.
```

Dans le premier cas, le guillemet indique que le mot CAP lui-même est l'argument de la procédure ECRIS.

Dans le deuxième cas, l'argument n'a pas de guillemet. On l'interprète par conséquent comme une procédure, qui fournit l'argument d'ECRIS.

Comment envisager DONNE
L'effet de la commande

```
DONNE "OISEAU "PIGEON
```

peut être envisagé comme suit. On donne un nom à une boîte: OISEAU. Le mot PIGEON est placé dans cette boîte.

Après cela, l'opération CHOSE a l'effet suivant:

```
CHOSE "OISEAU
   produit PIGEON
```

Les deux points ressemblent à CHOSE; par exemple,

```
:OISEAU
   produit PIGEON
```

On peut envisager cette situation de diverses manières:

OISEAU est une variable; PIGEON est sa valeur.
OISEAU est un nom de PIGEON.
OISEAU est lié à PIGEON.
PIGEON est la chose de OISEAU.

On peut modifier ce qui se trouve dans la boîte; si vous tapez

```
DONNE "OISEAU "MOINEAU
```

la boîte OISEAU contient le mot MOINEAU au lieu du mot PIGEON.

Comment envisager les procédures que vous définissez et leurs arguments.
Les procédures que vous définissez peuvent avoir des arguments. Lorsque vous définissez une procédure, la ligne titre spécifie le nombre des arguments, et donne à chacun d'eux un nom. Les arguments sont placés dans les boîtes indiquées sur la ligne titre de la définition de la procédure, dans l'ordre où ils se présentent. Par exemple, la procédure suivante fait tracer divers polygones par la Tortue, selon la valeur des arguments utilisés.

```
POUR POLY :COTE :ANGLE
AVANCE :COTE
DROITE :ANGLE
POLY :COTE :ANGLE
FIN
```

Les valeurs que vous choisissez pour vos arguments - par exemple, 50 et 90 - sont placées dans les boîtes indiquées. Le premier argument est toujours placé dans la boîte COTE; le deuxième argument est toujours placé dans la boîte ANGLE.
La première variable, COTE, contrôle la taille de la figure. La deuxième variable, ANGLE, contrôle la forme de la figure. POLY 50 90 fait tracer un carré à la Tortue; POLY 50 144 lui fait tracer un pentagramme (étoile); POLY 50 120 lui fait tracer un triangle.

Si les boîtes des arguments contiennent déjà des objets

lorsque l'on exécute la procédure, ces objets sont retirés, mais mis de côté. Il sont rétablis à la fin de l'exécution de la procédure. Autrement dit, la procédure emprunte la boîte, et lui rend sa condition d'origine une fois qu'elle a fini. C'est là ce que l'on signifie lorsqu'on dit qu'une variable est locale dans la procédure où elle se trouve.

Par exemple:

```
?DONNE "SON "GRONDEMENT
?ECRIS :SON
GRONDEMENT
```

Maintenant, vous exécutez une procédure appelée ANIMAL; sa variable argument est SON.

```
POUR ANIMAL :SON
SI :SON = "MIAOU [EC "CHAT STOP]
SI :SON = "OUAH [EC "CHIEN STOP]
EC [JE NE SAIS PAS]
FIN
```

Si nous tapons ANIMAL "OUAH, le mot OUAH est placé dans la boîte SON pour l'exécution de la procédure. Après la fin de l'exécution, le mot GRONDEMENT est remis dans la boîte SON:

```
?DONNE "SON "GRONDEMENT
?ECRIS :SON
GRONDEMENT
?ANIMAL "OUAH
CHIEN
?ECRIS :SON
GRONDEMENT
```

Une autre manière de considérer les procédures

Quand on utilise une procédure, on peut également dire que l'on appelle cette procédure. Les procédures que vous définissez appellent d'autres procédures. Par exemple, la procédure POLY ci-dessus appelle d'autres procédures (AVANCE et DROITE).

POLY s'appelle également elle-même. On dit qu'il s'agit d'un appel récursif.

Examen des caractères spéciaux

Le guillemet, ", utilisé immédiatement avant un mot, indique que celui-ci est considéré comme un mot et non comme le nom d'une procédure ou la valeur d'une variable.

Les deux points, :, utilisé immédiatement devant un mot, indique que le mot doit être considéré comme le nom d'une variable; il produit la valeur de cette variable.

Les crochets carrés, [], servent à enfermer une liste. Le crochet gauche ([) se tape comme SHIFT-N, le crochet droit (]) comme SHIFT-M.

Les parenthèses, (), sont nécessaires pour assurer des groupements qui ne sont pas les groupements normaux de Logo, et pour faire varier le nombre d'arguments pour certaines procédures. Ces deux utilisations sont décrites ci-dessus.

La barre inverse, \, déclare à Logo qu'il faut interpréter le caractère suivant littéralement, comme un caractère, au lieu de lui conserver un sens spécial qu'il pourrait avoir. Par exemple, si vous voulez utiliser 3[A]B comme un mot, il vous faut taper 3\[A\]B pour éviter que Logo n'attribue aux crochets leur interprétation habituelle, savoir l'enveloppe d'une liste. Il faut utiliser la barre inverse \ avec [,(,],),=,-,*,/,=,<,>, et la barre inverse elle-même.

Etant donné que la clavier de l'Apple ne contient pas de barre inverse, on tape CTRL-Q.

Il vous faut un système Apple II de 48K, avec une carte de langage (carte mémoire 16K) dans le port 0, un contrôleur de disque à 16 secteurs dans le port 6, au moins un lecteur de disque connecté au contrôleur, et un récepteur télé couleur ou noir et blanc. La mise en place du matériel Apple est traité dans les manuels Apple II.

Vous trouverez dans le nécessaire Logo une Disquette Logo. Il vous faut mettre cette disquette dans le lecteur de disque. Pour placer la disquette ouvrez la porte du lecteur en la soulevant vers l'extérieur, afin de libérer la fente. Tenez la disquette de telle sorte que l'étiquette soit vers le haut, glissez-la dans le lecteur, refermez la porte. Le lecteur est alors prêt à transférer l'information de la disquette à l'ordinateur.

Mettez la télé en marche. Puis pour lancer Logo, actionnez l'interrupteur de l'Apple qui se trouve à l'arrière de l'ordinateur sur votre gauche.

La lumière du lecteur de disque s'allume. Après un moment elle s'éteint et si tout va bien, vous devriez bientôt voir sur l'écran un message qui vous dit:

APPLE LOGO (c) 1982 LCSI

Vous n'avez pas besoin de disquette fichier pour utiliser ce manuel ou pour utiliser Logo. Si vous n'avez pas de disquette fichier, tapez simplement RETOUR et continuez.

Après que vous aurez pressé RETOUR, Logo ramènera le fichier STARTUP, s'il existe.*

*Si vous pressez CTRL-G au lieu de RETOUR, Logo ne ramène pas le fichier STARTUP.

A présent vous verrez à l'écran un message qui dit:

BIENVENUE A LOGO
?

Sous le message vous voyez un point d'interrogation ? suivi d'un rectangle clignotant.

Le "?" est le symbole d'invite. Lorsque le point d'interrogation apparaît sur l'écran, Logo attend que vous tapiez quelquechose au clavier. Lorsque vous tapez quelquechose en réponse au symbole d'invite, on dit que vous êtes au niveau supérieur.

Le rectangle clignotant est le curseur. Il apparaît lorsque Logo désire que vous tapiez quelque chose et indique la position où apparaîtra le caractère tapé.

Les touches spéciales au clavier

La touche <—
La touche <— permet d'effacer le caractère qui précède le curseur. Vous trouverez dans votre système Logo un autocollant marqué DELETE (efface) à coller sur cette touche pour vous en souvenir.

La touche RETOUR (RETURN)
La touche RETOUR permet de dire à Logo: "A présent fais ce que je viens d'écrire."

La touche "SHIFT"
Si cette touche est pressée en même temps qu'une autre, celle-ci peut changer de signification. Par exemple, pendant que la touche "SHIFT" est enfoncée, appuyez sur N. Logo imprime [sur l'écran. Nous écrirons cette combinaison de deux touches SHIFT-N. SHIFT-M est un]. La plupart de ces symboles sont inscrits dans la portion supérieure des touches du clavier de l'Apple.

Vérifiez que SHIFT-M et SHIFT-N sont marqués] et [sur le devant des touches M et N. Si ce n'est pas le cas, cherchez dans votre nécessaire Logo les autocollants qui vous permettront de les marquer.

La barre d'espace
Cette barre imprime un caractère invisible: l'espace. Logo utilise les espaces pour séparer les mots.

La touche CTRL
La touche CTRL peut transformer des touches de caractère en touches de commande. Elle s'utilise comme la touche "SHIFT". Si on la presse seule rien ne se passe, mais si on enfonce une autre touche pendant qu'elle est enfoncée il peut se passer quelque chose.

Par exemple la combinaison CTRL-G ordonne à Logo d'arrêter ce qu'il est en train de faire.

Comment formater une disquette
Pour sauvegarder vos procédures et autres objets Logo, il vous faut une disquette Apple formatée pour écrire. Si vous n'en avez pas, vous pouvez en faire une.

Vous pouvez formater une disquette au moyen du DOS3.3 SYSTEM MASTER de l'Apple. Placez la disquette dans le lecteur de disque et lancez celui-ci en remettant votre

Apple en marche. Lorsque le voyant du lecteur de disque s'éteint, retirez la disquette SYSTEM MASTER et remplacez-la par la disquette vierge que vous voulez formater.

Ensuite tapez

INIT HELLO

Lorsque le voyant du lecteur de disque s'éteint, la disquette sera formatée. Vous pouvez le vérifier en tapant

CATALOG

Vous devriez voir afficher le nom du fichier HELLO.

Ensuite placez la disquette Logo dans le lecteur de disque. Refaites la mise en marche de l'Apple. Lorsque Logo écrit

APPLE LOGO
(c) 1982 LCSI

enfoncez la touche RETOUR. N'enlevez pas encore la disquette Logo du lecteur. Donnez à Logo le temps de ramener le fichier STARTUP de la disquette.

Lorsque Logo écrit

BIENVENUE À LOGO

retirez la disquette Logo et insérez votre disquette de fichier. Ensuite tapez

SAUVE "STARTUP "AIDES

maintenant vous avez votre propre disquette de fichier, avec son fichier STARTUP.

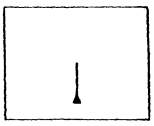
Chaque fois que vous utilisez une primitive qui fait intervenir la Tortue, Logo fait apparaître l'écran graphique, avec la Tortue.*

Outre les primitives décrites dans ce chapitre, les primitives suivantes relèvent du graphique Tortue: PLEINECRAN, MIXECRAN, TEXTECRAN; elles sont toutes décrites au chapitre 10.

L'écran graphique est effacé dès qu'on utilise l'éditeur Logo ou n'importe quelle commande de fichier.

RECULE
RE

RECULE distance (commande) abréviation: RE
Déplace la Tortue vers l'arrière du nombre de pas indiqué par distance. Le cap ne change pas.



RE 70

FOND

FOND (opération)
Retourne un nombre représentant la couleur du fond:
0 noir
1 blanc
2 vert
3 violet
4 orange
5 bleu
6 noir (pour télé noir & blanc)

NETTOIE

NETTOIE (commande)
Efface l'écran graphique sans affecter la Tortue.

NETTOIE

VIDECRAN
VE

VIDECRAN (commande) abréviation: VE
Efface l'écran graphique, remet la Tortue à la position [0 0] (centre de l'écran, appelée position origine) et fixe le cap de la Tortue à 0 (nord).

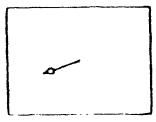
VE

POINT

POINT position (commande)
Dessine un point de la couleur en cours à la position désignée, sans bouger la Tortue ni tracer de ligne, même si le crayon est baissé.

Exemple:

POINT [100 0] place un point au milieu du bord droit de l'écran.



POINT [100 0]

BARRIERE

BARRIERE (commande)
Enferme la Tortue dans un champ limité aux bords de l'écran. Toute tentative de faire franchir ces limites à la Tortue provoquera l'apparition du message: "TORTUE EST HORS DES LIMITES". Un message d'erreur apparaît également si vous utilisez BARRIERE quand la Tortue est hors de l'écran.
(Voir aussi FENETRE et ENROULE).

Exemple:

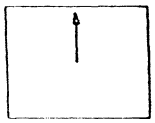
```
BARRIERE
VE
DR 5
AV 500
```
provoque l'apparition du message:
"TORTUE EST HORS DES LIMITES"

AVANCE
AV

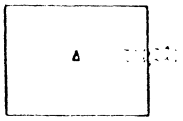
AVANCE distance (commande) abréviation: **AV**
Déplace la Tortue sur son cap, du nombre de pas indiqué par distance.

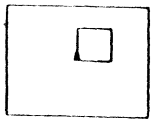
Exemples:

AV 70



```
POUR CARRE :COTE
REPETE 4 [AV :COTE DR 90]
FIN
```





CARRE 30

CAP

CAP (opération)
Retourne le cap de la Tortue qui est un nombre décimal supérieur ou égal à 0 et inférieur à 360. Logo compte les angles comme sur la boussole, où 0 est le nord, 90 l'est, 180 le sud et 270 l'ouest. A la mise en route de Logo, la Tortue a un cap 0 (vers le haut).

Exemple:

```
SI CAP = 180 [EC[VOUS ALLEZ DROIT AU SUD]]
```

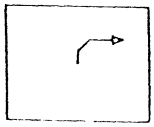
CACHETORTUE
CT

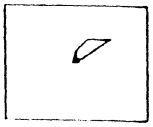
CACHETORTUE (commande) abréviation: **CT**
Rend la Tortue invisible (celle-ci dessine plus rapidement quand elle est cachée).

CACHETORTUE

CENTRE

CENTRE (commande)
Amène la Tortue au centre de l'écran et fixe le cap à 0. Cette instruction est équivalente à: FIXEPOS [0 0] FIXECAP 0 si le crayon est baissé, la Tortue dessinera son chemin jusqu'à l'origine.





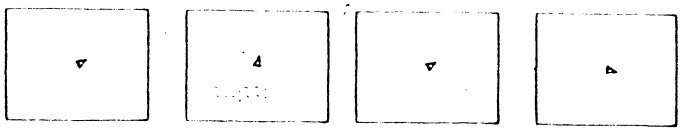
CENTRE

GAUCHE
GA

GAUCHE degrés (commande) abréviation: **GA**
Fait tourner la Tortue vers la gauche (sens contraire aiguille) du nombre de degrés indiqués. Il y a erreur commise si degrés > 4.19E6.

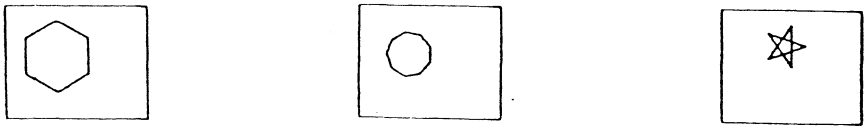
Exemples:

GAUCHE 45 (la Tortue tourne de 45 degrés vers la gauche)
GAUCHE -45 (la Tortue tourne de 45 degrés vers la droite)



La procédure POLY dessine les figures ci-dessous.

```
POUR POLY :COTE :ANGLE
AVANCE :COTE
GAUCHE :ANGLE
POLY :COTE :ANGLE
FIN
```



POLY 70 60 POLY 30 40 POLY 80 144

CRAYON

CRAYON (opération)
Retourne une liste de 2 mots décrivant l'état courant du crayon de la Tortue. Le premier mot peut être: BAISSECRAYON, EFFACECRAYON, LEVECRAYON ou INVERSECRAYON, le deuxième est un nombre exprimant le code de la couleur du crayon. Au démarrage de Logo, CRAYON retourne [BAISSECRAYON 1].

La liste retournée par crayon peut être utilisée comme information par FIXECRAYON (voir exemple dans FIXECRAYON).

COULEURCRAYON

COULEURCRAYON (opération)
Retourne un nombre représentant le code de la couleur courante:
0 noir
1 blanc
2 vert
3 violet
4 orange
5 bleu

Au départ de Logo, COULEURCRAYON est 1.

BAISSECRAYON
BC

BAISSECRAYON (commande) abréviation: BC
Place le crayon de la Tortue en position écriture. Dans

des lignes de la couleur
à son crayon baissé.

BC AV 100

GOMMECRAYON
GC

GOMMECRAYON (commande) abréviation: **GC**
Le crayon de la Tortue se transforme en gomme. La Tortue efface toute ligne sur laquelle elle passe. Pour changer cet état faites BAISSECRAYON ou LEVECRAYON.

GC AV 100

INVERSECRAYON
IC

INVERSECRAYON (commande) abréviation: **IC**

Le crayon de la Tortue se transforme en inverseur de couleur. Quand la Tortue se déplace, elle intervertit la couleur du crayon et celle du fond. Elle trace des lignes là ou il n'y en a pas et les efface là où elles existent. Les résultats de cette inversion sont complexes; ils dépendent de la couleur du fond, de celle du crayon, du caractère horizontal ou vertical des lignes. On obtient les meilleurs résultats sur un fond noir.

IC AV 100

LEVECRAYON
LC

LEVECRAYON (commande) abréviation: **LC**
Le crayon est levé: quand la Tortue se déplace, elle ne dessine pas.

LC AV 100

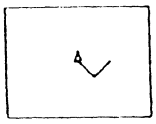
POS:

POS (opération)
Retourne les coordonnées de la position courante de la

[illegible] [x y]. Au départ de [illegible] [illegible] donne [0 0].

Exemple:

```
POUR LETTREVAI
DONNE "SAUVEPOS POS
VAI
LEVECRAYON
FIXEPOS :SAUVEPOS
FIN

POUR VAI
DR 135 AV 20
GA 90 AV 20
GA 45
FIN
```



LETTREVAI

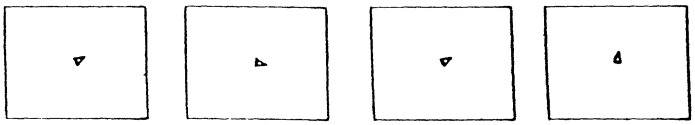
LETTREVAI appelle la procédure VAI et remet la Tortue à la position à laquelle elle se trouvait avant l'appel de LETTREVAL.

DROITE
DR

DROITE degrés (commande) abréviation: DR
Fait tourner la Tortue vers la droite (sens des aiguilles d'une montre) du nombre de degrés indiqué. Il y a erreur si degrés > 4.19E6.

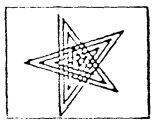
Exemples:

DROITE 45 (la Tortue tourne de 45 degrés vers la droite)
DROITE -45 (la Tortue tourne de 45 degrés vers la gauche)



DR 45 DR -45

```
POUR SPI :COTE :ANGLE :INC
AV :COTE :DR :ANGLE
SPI :COTE+:INC :ANGLE :INC
FIN
SPI 5 144 3
```



ECHELLE

ECHELLE (opération)

Retourne le rapport d'aspect, nombre décimal qui représente le rapport entre un pas vertical et un pas horizontal de la Tortue. Au départ ECHELLE vaut 0.8. (Voir aussi FIXECHELLE).

FIXEFOND

FIXEFOND codecouleur (commande) abréviation: FFD

FFD Fixe la couleur du fond à celle représentée par le nombre codecouleur, où codecouleur peut être:

0 noir
1 blanc
2 vert
3 violet
4 orange
5 bleu
6 noir

Remarquez que 0 et 6 représentent tous deux du noir; 6 est recommandé pour l'utilisation avec des écrans noir et blanc: il donne des lignes plus fines. (Voir exemple à FOND).

Il existe des limitations inévitables losqu'on dessine en couleur sur un fond de couleur. Le crayon noir ou blanc n'a pas de problèmes, quelle que soit la couleur du fond; de même, il n'y a pas de problème pour un crayon de couleur sur un fond noir ou blanc. Si vous essayez un tracé vert ou violet sur un fond orange ou bleu, ou inversement, voici ce qui se passe:

fond orange ou bleu
vert devient orange
violet devient bleu

fond vert ou violet
orange devient vert
bleu devient violet

Si vous changez le fond après avoir tracé des lignes de couleur, votre dessin risque d'être altéré.

FIXECAP
FCAP

FIXECAP degrés (commande) abréviation: FCAP

Fait tourner la Tortue de telle sorte que son cap ait la valeur indiquée par degrés (n'importe quel nombre décimal inférieur à 4.19E6). Les nombres positifs définissent une direction dans le sens des aiguilles d'une montre et les nombres négatifs dans le sens contraire aiguille à partir du nord. Remarquez que DROITE et GAUCHE définissent

[illegible] FIXECAP [illegible]fait un [illegible], [illegible] doit le cap courant de la Tortue.

Exemple:

FIXECAP 45 (dirige la Tortue vers le nord-est)
FIXECAP -45 (dirige la Tortue vers le nord-ouest)

FCAP 45 FCAP -45

FIXECC
FCC

FIXECC codecouleur (commande) abréviation: **FCC**
Fixe la couleur du crayon à la couleur définie par codecouleur où codecouleur est un des nombres:

0 noir
1 blanc
2 vert
3 violet
4 orange
5 bleu
6 noir

Si la couleur du crayon n'est pas exactement celle que vous désiriez, essayez d'abord d'ajuster votre TV (couleur et contraste). De toute manière quand 2 lignes horizontales de couleurs différentes se touchent, l'une d'elles peut être d'une mauvaise couleur. Vous n'y pouvez rien.

Pour plus d'information sur les relations entre les couleurs du crayon et du fond, voir FIXEFOND.

FIXECRAYON

FIXECRAYON paire (commande)
Fixe l'état du crayon à la valeur indiquée par la liste paire. La forme de l'information est la même que celle que retourne CRAYON (voir CRAYON). Le premier mot doit être: BAISSECRAYON, EFFACECRAYON, LEVECRAYON ou INVERSECRAYON; le deuxième est un nombre représentant le codecouleur du crayon.

Exemple:

FIXECRAYON [LC 3] est équivalent à:
LEVECRAYON
FIXECC 3

Dans le programme suivant, la procédure DESSIN dessine 3 côtés d'un carré en 3 couleurs différentes. Elle laisse le crayon levé, et la couleur du crayon bleue. La procédure

RESTAURE ... de vos procédure et s'assure que le crayon est remis dans l'état où il se trouvait auparavant.

```
POUR RESTAURE :PROCEDURE
DONNE "SAUVECRAYON  CRAYON
EXECUTE PH :PROCEDURE
FIXECRAYON :SAUVECRAYON
FIN
```

```
POUR DESSIN
COTE 2
COTE 3
COTE 5
LEVECRAYON AV 50
FIN
```

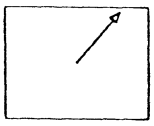
```
POUR COTE :COULEUR
FIXECC :COULEUR
AV 50 DR 90
FIN
```

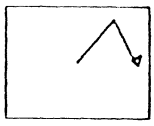
FIXEPOS
FPOS

FIXEPOS position (commande)
Place la Tortue à position (voir POS)

Exemple:

FIXEPOS [100 0] place la Tortue au milieu du bord droit de l'écran.





FPOS [100 0]

FIXECHELLE

FIXECHELLE n (commande)
Fixe le rapport d'aspect (rapport entre la taille d'un pas vertical et celle d'un pas horizontal de la Tortue) à n, et modifie YCOR en conséquence.

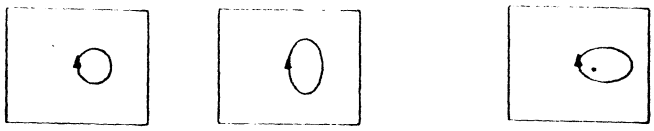
Exemple:

FIXECHELLE .5 donne à la longueur de chaque pas vertical de la Tortue une valeur moitié de celle d'un pas horizontal.

FIXECHELLE sert à deux fins. La première, si les "carrés" ressemblent à des rectangles et les "cercles" à des ellipses à l'écran, il permet de corriger l'impression visuelle (pour la plupart des écrans, 0.8 est le bon rapport). La seconde vous permet de comprimer ou d'étirer les dessins de la Tortue. Par exemple vous pouvez

[illegible] un cercle pour dessiner des ellipses.

```
POUR CERCLE :RAYON
REPETE 60 [AV :RAYON * 3.14159 / 30!
DR 6]
FIN

POUR ELLIPSE :HORIZ :VERT
FIXECHELLE .8 * :VERT / :HORIZ
CERCLE :HORIZ
FIN
```



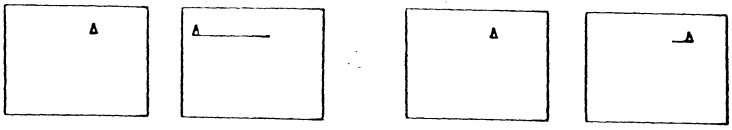
CERCLE 25 ELLIPSE 25 40 ELLIPSE 40 25

FIXEX
FX

FIXEX x (commande) abréviation: **FX**
Déplace la Tortue horizontalement jusqu'au point d'abscisse x (l'ordonnée y reste inchangée).

Exemple:

FIXEX -100 déplace la Tortue horizontalement jusqu'au bord gauche de l'écran.



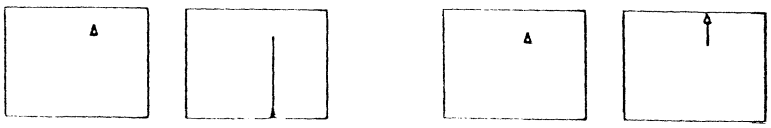
FX -100 FX 2* XCOR

FIXEY
FY

FIXEY y (commande) abréviation: **FY**
Déplace la Tortue verticalement jusqu'au point d'ordonnée y (l'abscisse x ne change pas).

Exemple:

FIXEY -100 déplace la Tortue verticalement jusqu'au bas de l'écran.



FY -100 FY 2* YCOR

[illegible]P [illegible]
[illegible] VRAI si la Tortue est visible, FAUX sinon.

MONTRETORTUE
MT

MONTRETORTUE (commande) abréviation: **MT**
Rend la Tortue visible. (Voir CACHETORTUE).

MONTRETORTUE

VERS

VERS position (opération)
Retourne un cap qui est celui que la Tortue devrait adopter pour pointer vers la position indiquée.

Exemple:

FIXECAP VERS [20 10] Dirige la Tortue dans la direction de la position [20 10]

FCAP VERS [20 10]

FENETRE

FENETRE (commande)
Rend le champ de la Tortue illimité; ce que vous voyez à l'écran est la portion du champ que vous apercevez au travers d'une fenêtre dont le centre serait le centre de votre écran. Quand la Tortue échappe à votre regard, elle continue à se déplacer hors de votre vue. L'écran est de 240 pas de Tortue en hauteur (si l'échelle est 0.8) et 280 pas de large. Le champ complet de la Tortue est de 40960 pas en hauteur* et 32768 en largeur. (Voir aussi BARRIERE et ENROULE).

Exemple:

```
?FENETRE
?VE DR 5
?AV 500
?EC POS
43.5779 498.097
```

*Selon la valeur de [illegible]

ENROULE

ENROULE (commande)
Enroule le champ de la Tortue autour des bords de l'écran: si la Tortue traverse un bord elle réapparaît sur le côté opposé. Elle n'échappe jamais aux limites visibles:

Exemple:

```
?ENROULE
?VE DR 5
?AV 500
?EC POS
43.5779 18.0973
```

XCOR

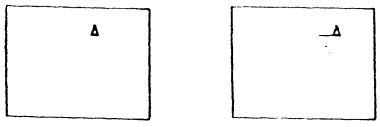
XCOR (opération)
Retourne l'abscisse de la position courante de la tortue.

Exemples:

```
ECRIS XCOR
0
```

```
EC XCOR
0
```

FIXEX 2*XCOR déplace la tortue horizontalement jusqu'à un point dont l'abscisse est double de celle de la position précédente.



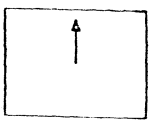
```
FX 2*XCOR
```

YCOR

YCOR (opération)
Retourne l'ordonnée de la position courante de la tortue.

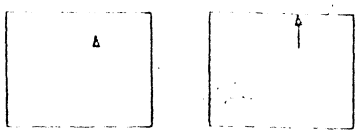
Exemple:

```
ECRIS YCOR
100
```



```
EC YCOR
100
```

FY 2*YCOR [illegible] la tortue vertic[illegible]ement jusqu'à un point dont l'ordonnée est double de celle de la position précédente.



FY 2*YCOR

Chapitre 2

Il existe dans LOGO 2 types d'objets: les mots et les listes, il existe des primitives pour les rassembler, les démonter, les examiner.

Un mot est composé de caractères.

BONJOUR
X
314
3.14
R2D2
COCHOND'INDE
COCHON.D'INDE
COCHON-D'INDE (à taper sous la forme COCHON\-D'INDE)
QUI?
!COMMENT!

sont tous des mots. Chaque caractère est un élément du mot. Le mot COCHON.D'INDE contient 13 éléments.

C O C H O N . D ' I N D E

Un mot est généralement délimité par des espaces. Cela signifie qu'il y a un espace avant (à moins que le mot ne soit précédé de : ou de ") et un espace après; Cela permet de distinguer le mot du reste de la ligne. Il existe certains autres caractères délimiteurs;

[ ] ( ) = < > + - * / .

pour considérer chacun de ces caractères comme un caractère alphabétique normal, faites le précéder d'une barre inverse "\" (tapée au clavier par CTRL-Q).

Exemple:

?EC "COCHON\-D'INDE
COCHON-D'INDE
(Voir appendice 1 pour plus de détails)

Remarquez que " n'est pas un délimiteur de mots.

Une liste est composé d'objets de LOGO, chacun peut être un mot ou une autre liste. On signale que quelque chose est une liste en l'entourant de crochets. Voici des listes:

[BONJOUR, COMMENT ALLEZ VOUS?]

```
[X Y Z]
[SALUT]
[[MAISON HOUSE] [FENETRE WINDOW] [CHIEN DOG]]
[ALCAPONE [C3PO R2D2] [TINTIN] [D'ARTAGNAN]]
[1 [1 2] [17 [17 2]]]
[ ]
```

La liste [BONJOUR, COMMENT ALLEZ VOUS?] contient 4 éléments:

BONJOUR,
COMMENT
ALLEZ
VOUS?

Remarquez que la liste: [1 [1 2] [17 [17 2]]] ne contient que 3 éléments, le second et le troisième étant eux-mêmes des listes.

Premier élément: 1
Deuxième élément: [1 2]
Troisième élément: [17 [17 2]]

La liste [ ] qui ne contient aucun élément est la liste vide. Il existe aussi un mot vide. Vous le tapez au clavier en tapant ".
(Voir l'entrée VIDEP pour des exemples de liste vide et mot vide)

ASCII

ASCII caractère (opération)
Retourne le code ASCII de caractère. (L'appendice K donne une table de tous les codes ASCII). Si l'information donnée contient plus d'un caractère, ASCII n'opère que sur le premier. (voir aussi CAR)

Exemple:

ASCII "B retourne 66.

La procédure CODESECRET définit un nouveau mot en utilisant le code chiffré de César (ajouter 3 à chaque lettre).

```
POUR CODESECRET :MT
SI VIDEP :MT [RETOURNE "]
RET MOT CODE PREMIER :MT!
CODESECRET SP :MT
FIN

POUR CODE :LETTRE
DONNE "NUMERO (ASCII :LETTRE) + 3
SI :NUMERO > ASCII "Z [DONNE "NUMERO :NUMERO - 26]
```

```
[illegible]
FIN

?EC CODE "CHAT
F K D W

?EC CODE "CRAYON
F U D B R Q
```

SAUFPREMIER
SP

SAUFPREMIER objet (opération) abréviation: SP
Retourne la liste ou le mot objet sauf son premier élément. SAUFPREMIER d'une liste vide ou d'un mot vide est une erreur.

Exemple:

```
SAUFPREMIER [ALBERT EINSTEIN]
   retourne [EINSTEIN]
SP "TROIS
   retourne ROIS

SP [TROIS]
   retourne [] (liste vide)

SP [LES TROIS]
   retourne [TROIS]

SP "
   est une erreur

POUR TRIANGLE :OBJET
SI VIDEP :OBJET [STOP]
EC :OBJET
TRIANGLE SAUFPREMIER :OBJET
FIN

?TRIANGLE "TROIS
TROIS
ROIS
OIS
IS
S
```

SAUFDERNIER

SAUFDERNIER objet (opération)
Retourne la liste ou le mot objet sauf son dernier élément.

Exemple:

```
SAUFDERNIER [ALBERT EINSTEIN]
   retourne ALBERT

SAUFDERNIER "HEURES
   retourne HEURE
```

```
SD [ ... S]
   ... []

SD "
   est une erreur

SD [ ]
   est aussi une erreur
```

L'entrée de la procédure suivante est un verbe en -er.

```
POUR CONJUGUE :MOT
EC PH [JE] SD :MOT
EC PH [NOUS] MOT SD SD :MOT "ONS

CONJUGUE "CHANTER
JE CHANTE
NOUS CHANTONS
```

CAR

CAR n (opération)
Retourne le caractère dont le code ASCII est n (voir en appendice K la table des codes ASCII). Il y a erreur si n n'est pas un code ASCII.

Les Caractères peuvent être normaux (blanc sur fond noir) en vidéo inverse (noir sur fond blanc), ou clignotants.

Les codes ASCII sont organisés de la manière suivante:

0- 31	caractères de contrôle (combinaisons avec CTRL)
32- 47	ponctuation
48- 57	chiffres
58- 63	ponctuation
64- 90	majuscules
91- 96	ponctuation
97-122	minuscules (majuscules sur APPLE II)
123-127	ponctuation
128-154	vidéo inverse, majuscules
155-191	vidéo inverse, chiffres et ponctuation
192-218	clignotants, majuscules
219-255	clignotants, chiffres et ponctuation.

Pour transformer un caractère normal en caractère vidéo-inverse utilisez l'expression:

```
CAR 128 + RESTE ASCII :CARACTERE 64
```

Pour le transformer en caractère clignotant

```
CAR 192 + RESTE ASCII :CARACTERE 64
```

Exemples:

INVERSE et CLIGNOTE affichent un mot soit en vidéo

```
POUR INVERSE :MOT
TRANSFORMER :MOT 128
FIN

POUR CLIGNOTE :MOT
TRANSFORMER :MOT 192
FIN

POUR TRANSFORMER :MOT :NOMBRE
SI VIDEP :MOT [STOP]
TAPE CAR :NOMBRE + RESTE ASCII PREMIER :MOT 64
TRANSFORMER SP :MOT :NOMBRE
FIN

INVERSE "FROMAGE
FROMAGE ?
```

COMPTE

COMPTE liste (opération)
Retourne le nombre d'éléments de la liste

Exemple:

```
COMPTE [VOYEZ LE BRICK]
   retourne 3

COMPTE [[VOYEZ [LE BRICK] GEANT]
   retourne 2

?DONNE "CLASSE [FRANCOISE MICHEL ALAIN
GERARD SYLVAIN MARIE\-PAULE ISIDORE]
?EC COMPTE :CLASSE
7
```

La procédure suivante affiche un élément pris au hasard dans une liste:

```
POUR PIQUEHASARD :LISTE
EC ITEM (1 + HASARD COMPTE :LISTE) :LISTE
FIN

?PIQUEHASARD :CLASSE
MARIE-PAULE
```

VIDEP

VIDEP objet (opération)
Retourne VRAI si objet est le mot vide ou la liste vide, FAUX sinon.

Exemples:

```
VIDEP 3
   retourne FAUX
```

```
VIDEP SAUFPREMIER "CHAPEAU
   retourne FAUX

VIDEP SAUFDERNIER "U
   retourne VRAI

VIDEP SP [CHAPEAU]
   retourne VRAI

POUR CRIS :ANIMAUX :BRUITS
SI OU VIDEP :CRIS VIDEP :ANIMAUX!
[EC [VOILA, C'EST TOUT] STOP]
EC PH PREMIER :ANIMAUX PREMIER :CRIS
CRIS SP :ANIMAUX SP :BRUITS
FIN

?CRIS [CHIENS OISEAUX COCHONS] [ABOYER PEPIER
GROGNER]

CHIENS ABOYER
OISEAUX PEPIER
COCHONS GROGNER

POUR ENVERS :CHOSE
SI VIDEP :CHOSE [EC [] STOP]
TAPE DERNIER :CHOSE
SI LISTEP :CHOSE [TAPE CAR 32] (espace)
ENVERS SD :CHOSE
FIN

ENVERS "ELEPHANT
TNAHPELE
```

EGALP

EGALP objet1 objet2 (opération)
Retourne VRAI si chose 1 et chose 2 sont des nombres égaux, ou des mots ou listes identiques.
Synonyme du symbole = (voir liste des formes infixes à la fin du chap. 4)

Exemples:

EGALP "[] retourne FAUX (le mot vide et la liste vide ne sont pas identiques)

La procédure suivante indique si le premier argument (un caractère) est un élément du second (un mot).

```
POUR PRESENCE :CAR :MOT
SI VIDEP :MOT RET "FAUX
SI EGALP :CAR PREMIER :MOT RET "VRAI
RETOURNE PRESENCE :CAR SAUFPREMIER :MOT
FIN

EC PRESENCE "H "THE
```

```
SI
EC PRÉSENCE "SAUF "SAUFPREMIER
FAUX
EC PRÉSENCE "SAUF "SAUFPREMIER
FAUX
```

PREMIER

PREMIER objet (opération)
Retourne le premier élément de l'objet. PREMIER appliqué à la liste ou au mot vide donne une erreur.
PREMIER d'un mot est un caractère.
PREMIER d'une liste peut être un mot ou une liste.

Exemples:

```
PREMIER [CAILLOU HIBOU JOUJOU]
   retourne CAILLOU

PREMIER "PALAIS
   retourne P

PREMIER [JOUJOU]
   retourne JOUJOU

PREMIER [[LE UN] [CHAT CHIEN RAT] [CRIE RIT COURT]]
   retourne [LE UN]

PREMIER "
   donne une erreur

PREMIER []
   donne aussi une erreur

POUR EPELLE :INFO
SI VIDEP :INFO [STOP]
EC PREM :INFO
EPELLE SP :INFO
FIN

?EPELLE "SOURIS
S
O
U
R
I
S

?EPELLE [A VOS SOUHAITS]
A
VOS
SOUHAITS
```

METD

METD objet liste (opération)
(forme contractée de "METs au Début")

[illegible] retourne en ajoutant objet au début de liste. (cf. LISTE pour comparaison de METD avec les autres primitifs combinant mots et listes).

Exemples:

METD "SOURIS [CHIEN CHAT]
  retourne [SOURIS CHIEN CHAT]

METD [LE UN] [BOL VERRE]
  retourne [[LE UN] BOL VERRE]

METD "A []
  retourne [A]

ITEM

ITEM n liste (opération)
Retourne le nième élément de liste. Il y a erreur si n est supérieur à la longueur de liste ou si liste est la liste vide.

Exemples:

?DONNE "ANIMAUX [CHIEN CHAT GIRAFE]
?EC ITEM 3 :ANIMAUX
GIRAFE
?EC ITEM 1 :ANIMAUX
CHIEN

DERNIER

DERNIER objet (opération)
Retourne le dernier élément d'objet. DERNIER appliqué au mot vide ou à la liste vide donne une erreur.

Exemples:

DERNIER [MARIE SEBASTIEN ONESIME]
  retourne ONESIME

DERNIER "CANNELLE
  retourne E

DERNIER [CANNELLE]
  retourne CANNELLE

DERNIER [[LE UN] SOLDAT VA [PARTIR SAUTER DORMIR]]
  retourne [PARTIR SAUTER DORMIR]

DERNIER "
  donne une erreur

DERNIER []
  donne aussi une erreur

POUR INVEPELLE :ENTR

```
[illegible]
EC [illegible]
?INVERSE "BONJOUR
FIN

?INVERSE "EDAGALF
F
L
A
G
A
D
A
```

LISTE

LISTE objet1 objet2 (opération)
(LISTE objet1 objet2 objet3...)
Retourne une liste dont les éléments sont objet 1, objet 2 etc...

Exemples:

```
LISTE "ROSE [TULIPE OEILLET]
   retourne [ROSE [TULIPE OEILLET]]

(LISTE "ROSE "TULIPE "OEILLET)
   retourne [ROSE TULIPE OEILLET]

LISTE [LA RAISON DU PLUS FORT][EST TOUJOURS LA
MEILLEURE]
   retourne [[LA RAISON DU PLUS FORT][EST
TOUJOURS LA MEILLEURE]]

LISTE "A[]
   retourne [A[]]
```

Si LISTE est utilisée avec un seul argument, les parenthèses autour de l'expression ne sont obligatoires que s'il suit autre chose sur la ligne sur laquelle elle figure.

```
?DONNE "ANIMAUX "CRAPAUDS
?MONTRE LISTE :ANIMAUX
[CRAPAUDS]

?MONTRE (LISTE :ANIMAUX) EC "RELAXE
[CRAPAUDS]
RELAXE
```

Le tableau ci-dessous montre les 4 primitives combinant listes et mots.

opération	argument 1	argument 2	valeur
METD	"VACHE	"CHEVAL	erreur
LISTE	"VACHE	"CHEVAL	[VACHE CHEVAL]
METF	"VACHE	"CHEVAL	erreur
PHRASE	"VACHE	"CHEVAL	[VACHE CHEVAL]
METD	"LOGO	[EST SUPER]	[LOGO EST SUPER]
LISTE	"LOGO	[EST SUPER]	[LOGO [EST SUPER]]
METF	"LOGO	[EST SUPER]	[EST SUPER LOGO]
PHRASE	"LOGO	[EST SUPER]	[LOGO EST SUPER]
METD	[LE CHAT]	[REGARDE MEDOR]	[[LE CHAT] REGARDE MEDOR]
LISTE	[LE CHAT]	[REGARDE MEDOR]	[[LE CHAT] [REGARDE MEDOR]
METF	[LE CHAT]	[REGARDE MEDOR]	[REGARDE MEDOR [LE CHAT]]
PHRASE	[LE CHAT]	[REGARDE MEDOR]	[LE CHAT REGARDE MEDOR]
METD	"ORDINATEUR	[]	[ORDINATEURS]
LISTE	"ORDINATEUR	[]	[ORDINATEURS []]
METF	"ORDINATEUR	[]	[ORDINATEURS]
PHRASE	"ORDINATEUR	[]	[ORDINATEURS]

LISTEP

LISTEP objet (opération)
Retourne VRAI si objet est une liste, FAUX sinon.

Exemples:

LISTEP 3
retourne FAUX

LISTEP [3]
retourne VRAI

LISTEP []
retourne VRAI

LISTEP "
retourne FAUX

LISTEP [A B C [D E][F[G]]]
retourne VRAI

LISTEP SP "CHOCOLAT
retourne FAUX

LISTEP SP [CHOCOLAT]
retourne VRAI

METF

METF objet liste (opération)
(forme contractée pour METs à la Fin)
Retourne une nouvelle liste obtenue en ajoutant objet à la fin de liste. Cf tableau sous LISTE pour une comparaison de METF avec les autres primitives combinant listes et mots.

METF "SOURIS [HAMSTER COCHON.D'INDE]
retourne [HAMSTER COCHON.D'INDE SOURIS]

METF [LE LA LES [CHAT GIRAFE]
retourne [CHAT GIRAFE [LE LA LES]]

METF "A []
retourne [A]

DERNIER METF "SOURIS [HAMSTER COCHON.D'INDE]
retourne SOURIS

La procédure suivante ajoute un mot à un dictionnaire français-espagnol:

```
POUR NOUVEAUMOT :INFO
DONNE "DICTIONNAIRE METF :INFO :DICTIONNAIRE
FIN
```

```
? DONNE "DICTIONNAIRE [MAISON CASA] [ESPAGNOL ESPANOL] [COMMENT COMO]

? MONTRE :DICTIONNAIRE
[[MAISON CASA] [ESPAGNOL ESPANOL] [COMMENT COMO]]

?NOUVEAUMOT [TABLE MESA]

?MONTRE :DICTIONNAIRE
[[MAISON CASA] [ESPAGNOL ESPANOL] [COMMENT COMO] [TABLE MESA]]
```

MEMBREP

MEMBREP objet liste (opération)
Retourne VRAI si objet est un élément de liste; FAUX sinon.

Exemple:

```
MEMBREP 3 [2 5 [3] 6]
   retourne "FAUX

MEMBREP 3 [2 5 3 6]
   retourne "VRAI

MEMBREP [2 5] [2 5 3 6]
   retourne "FAUX

MEMBREP "PIN "LAPIN donne une erreur

MEMBREP [MONACO LUXEMBOURG] [[MONACO LUXEMBOURG ] ANDORRE]
   retourne "VRAI
```

```
MEMBREP [MON LUX] [MON LUX AND]
   retourne "FAUX

MEMBREP SP "RAT [AR AS AT AU]
   retourne "VRAI
```

La procédure suivante détermine si l'entrée est une voyelle.

```
POUR VOYELLEP :LETTRE
RET MEMBREP :LETTRE [A E I O U]
FIN

EC VOYELLEP "F
FAUX
EC VOYELLEP "A
VRAI
```

NOMBREP

NOMBREP objet (opération)
Retourne VRAI si objet est un nombre, FAUX sinon.

Exemples:

```
NOMBREP 3
   retourne "VRAI

NOMBREP [3]
   retourne "FAUX

NOMBREP 3.14E23
   retourne "VRAI

NOMBREP []
   retourne "FAUX

NOMBREP "
   retourne "FAUX

NOMBREP SP 3165.2
   retourne "VRAI

NOMBREP SP "ELEPHANT
   retourne FAUX
```

PHRASE
PH

PHRASE objet1 objet2 (opération) abréviation: PH
(PHRASE objet1 objet2 objet3 ...)
Retourne une liste composée des mots contenus dans ses arguments.

Exemples:

```
PH "PAPIER "CAHIER
   retourne [PAPIER CAHIER]
```

```
PH [CAHIER] [DE CAHIER]
   retourne [CAHIER DE CAHIER]

PH "POMME [PÊCHE POIRE ABRICOT]
   retourne [POMME PÊCHE POIRE ABRICOT]

PH [LA RAISON DU PLUS FORT] [EST TOUJOURS LA
MEILLEURE]
   retourne [LA RAISON DU PLUS FORT EST
TOUJOURS LA MEILLEURE]

POUR FLATTER :QUI
EC PH [BONJOUR,] :QUI
EC []
EC PH :QUI [A DE LA CLASSE]
EC PH MOT :QUI :QUI [A DEUX FOIS PLUS DE CLASSE]

?FLATTER "JACQUES
BONJOUR, JACQUES
JACQUES A DE LA CLASSE
JACQUESJACQUES A DEUX FOIS PLUS DE CLASSE
```

Voir dans LISTE un tableau comparant PHRASE avec les autres primitives combinant mots et listes.

Autres exemples:

```
(PH "POMME "PÊCHE "POIRE)
   retourne [POMME PÊCHE POIRE]

(PH "MONET)
   retourne [MONET]

PH "MONET []
   retourne aussi [MONET]
```

Si PHRASE est utilisée avec un seul argument, les parenthèses autour de l'expression ne sont obligatoires que si autre chose suit l'expression sur la ligne qui la contient.

```
?DONNE "ANIMAUX "CHATONS
?MONTRE PH :ANIMAUX
[CHATONS]
?MONTRE (PH :ANIMAUX) EC "JOUER
[CHATONS]
JOUER
```

Comparez ce que retournent PHRASE et LISTE quand vous les appliquez à des arguments qui sont des listes de listes.

```
PH [LE CHIEN] [AIME [LES CARAMELS]]
   retourne [LE CHIEN AIME [LES CARAMELS]]
```

```
PRINT [LE CHIEN][AIME LES CARAMELS]
   retourne [[LE CHIEN][AIME LES CARAMELS]]
```

MOT

MOT mot1 mot2 (opération)
Retourne un mot composé des arguments mot1, mot2,...

Exemples:

```
MOT "COU "LEUR
   retourne COULEUR

MOT "APO "CALY "PSE
   retourne APOCALYPSE

MOT "COU [LEUR]
   donne une erreur

MOT "S "MILE
   retourne SMILE
```

MOTP

MOTP objet (opération)
Retourne VRAI si objet est un mot; FAUX sinon.

Exemples:

```
MOTP 3
   retourne VRAI

MOTP [3]
   retourne FAUX

MOTP "ZAP
   retourne VRAI

MOTP [SORT IE]
   retourne FAUX

MOTP []
   retourne FAUX

MOTP "
   retourne VRAI

MOTP SP "AUTO
   retourne VRAI

MOTP SP [AUTO]
   retourne FAUX
```

Un mot LOGO peut s'utiliser comme variable; une variable est une "boîte" qui contient un objet LOGO. Cet objet est appelé la valeur du mot. Ceci se réalise soit en utilisant DONNE ou NOMME, soit par les arguments d'une procédure.
(Voir le résumé et l'index pour une discussion des arguments).

LOCAL

LOCAL nom (commande)
(LOCAL nom1 nom2...)
Le ou les arguments de LOCAL ont une définition locale à la procédure à l'intérieur de laquelle LOCAL intervient. Cela signifie que chaque terme n'est accessible qu'à cette procédure et à celles qu'elle appelle. De ce point de vue ces arguments ressemblent à des arguments de la procédure.

Exemple:

```
POUR OUINON :QUESTION
LOCAL "REPONSE
EC :QUESTION
DONNE "REPONSE PREM LISLISTE
SI EGALP :REPONSE "OUI [RET "VRAI]
RET "FAUX
FIN

POUR SALUER
EC [QUEL EST VOTRE NOM COMPLET?]
DONNE "REPONSE LISLISTE
SI OUINON [AIMEZ VOUS VOTRE NOM?] [EC [C'EST
BIEN] [EC [C'EST DOMMAGE]]
EC PH [HEUREUX DE VOUS CONNAITRE, \ ]
:REPONSE
FIN

?SALUER
QUEL EST VOTRE NOM COMPLET?
HERMAN NIXON
AIMEZ VOUS VOTRE NOM?
NON
C'EST DOMMAGE
HEUREUX DE VOUS CONNAÎTRE, HERMAN NIXON
```

Imaginez ce qui se produirait si LOCAL avait été omis dans OUINON. Chaque procédure utilise une variable appelée REPONSE pour contenir la réponse de l'utilisateur. Si les variables n'étaient pas locales OUINON détruirait la valeur de REPONSE que SALUER s'attend à

```
?SALUER
QUEL EST VOTRE NOM COMPLET?
HERMAN NIXON
AIMEZ VOUS VOTRE NOM?
NON
C'EST DOMMAGE
HEUREUX DE VOUS CONNAÎTRE, NON
```

DONNE

DONNE nom objet (commande)
Donne le nom à la variable qui contient la valeur objet, autrement dit met la valeur objet dans la boîte nom.

Exemples:

```
DONNE "EMPLOI 259
?EC :EMPLOI
259
?DONNE "EMPLOI "SOUDEUR
?EC :EMPLOI
SOUDEUR
DONNE "SOUDEUR 32
?EC :SOUDEUR
32
?EC CHOSE :EMPLOI
32
DONNE :EMPLOI [HENRI MARTIN]
```

A ce point, :EMPLOI est SOUDEUR et CHOSE :EMPLOI est [HENRI MARTIN]

```
?EC "EMPLOI
EMPLOI
?EC :EMPLOI
SOUDEUR
?EC CHOSE "EMPLOI
SOUDEUR
?EC CHOSE :EMPLOI
[HENRI MARTIN]
```

NOMME

NOMME objet nom (commande)

```
?NOMME 259 "EMPLOI
?EC :EMPLOI
259
?NOMME "SOUDEUR "EMPLOI
?EC :EMPLOI
SOUDEUR
```

NOMME est équivalent à DONNE avec l'ordre des arguments inversé. Il place l'objet dans la boîte nom. Ainsi, NOMME "SOUDEUR "EMPLOI a le même effet que DONNE "EMPLOI "SOUDEUR.

Retourne VRAI si nom a une valeur (c'est à-dire si nom a une valeur), FAUX sinon.

Exemples:

```
?EC NOMP "ANIMAL
FAUX
?DONNE "ANIMAL "AARDVARK
?EC "ANIMAL
AARDVARK
?EC NOMP "ANIMAL
VRAI
```

La procédure INC ci-dessous, montre une utilisation de NOMP.

CHOSE

CHOSE nom (opération)
Retrouve l'objet contenu dans la boîte nom, c'est-à-dire la valeur de la variable nom. CHOSE "QQCH est équivalent à :QQCH

Exemple:

La procédure suivante incrémente (augmente de 1) la valeur d'une variable.

```
POUR INC :X
SI NON NOMP :X [STOP]
SI NOMBREP CHOSE :X [DONNE :X 1 + CHOSE :X]
FIN
```

Remarquez l'utilisation de DONNE :X plutôt que DONNE "X. En effet ce n'est pas X que l'on doit incrémenter. La valeur de X n'est pas un nombre, mais le nom d'une autre variable. C'est celle-ci qui doit être incrémentée.

```
?DONNE "TOTAL 7
?EC :TOTAL
7

?INC "TOTAL
?EC :TOTAL
8
```

On trouvera d'autres exemples à la rubrique DONNE deviendra ci-dessous.

LOGO reconnaît des nombres entiers et des nombres décimaux:
3 est un entier
3.14 et 3. sont des nombres décimaux.

Logo vous fournit des primitives qui vous permettent d'additionner, soustraire, multiplier et diviser les nombres. Vous pouvez déterminer sinus, cosinus, arctangentes et racines carrées, et aussi tester si un nombre est égal, inférieur ou supérieur à un autre nombre.

Certaines opérations arithmétiques (ENT, QUOTIENT, HASARD, RESTE, ARRONDIS) retournent toujours un entier, d'autres une décimal (ARCTAN, COS, SIN, RAC, /); certaines retournent des entiers si les arguments sont entiers, des décimaux sinon (+, -, *).

Ainsi 7 / 2 est 3.5 (décimal)
QUOTIENT 7 2 est 3 (entier)

3.5 + 6.5 est 10. (décimal), mais 7 + 3 est 10 (entier)
Remarquez que: 3 + 7. est 10. (décimal).

L'entier le plus grand que connaisse Logo est 2147483647, soit 2 - 1; le plus petit est 2147483648, soit —2 . Les nombres décimaux à plus de 6 chiffres sont convertis à la forme "scientifique" (notation exponentielle). Par exemple:

2E6 signifie 2 multiplié par 10 , soit 2 000 000;
2.59N4 signifie 2.59 multiplié par 10 , soit .0002.59
(N indique un exposant négatif).

Les valeurs des exposants peuvent être comprises entre -38 et +37. Les nombres décimaux à plus de 6 chiffres sont convertit par Logo sous la forme scientifique (notation exponentielle). Par exemple 2718281828459.045 est converti en 2.71828E12.

Addition, soustraction, multiplication et division ont une forme infixe. Dans ce cas l'infixe se place entre les arguments sur lesquelles il opère. Addition et multiplication disposent aussi d'une forme préfixe. Par exemple:

2 + 1 et SOMME 2 1 sont des expressions équivalentes.

En outre, la primitive EGALP, fréquemment utilisée en relation avec des opérations arithmétiques, est décrite au Chapitre 2 (mots et listes). Sa forme infixe "=" est

Formes préfixes

ARCTG

ARCTG n (opération)
Retourne l'arctangente de n

Exemples:

```
ARCTG 2
   retourne 63.4348

ARCTG 444
   retourne 89.8710
```

Les procédures suivantes définissent ARCSIN et ARCCOS

```
POUR ARCSIN :X
RET ARCTG :X /(RAC 1 - :X * :X).
FIN

POUR ARCCOS :X
RET ARCTG  (RAC 1 - :X * :X) / :X
FIN
```

COS

COS n (opération)
Retourne le cosinus de n degrés. Il y a erreur si n > 4.19E6

Exemples:

```
COS 60
   retourne .5

COS 30
   retourne .866025
```

Voici une définition de la fonction tangente:

```
POUR TG :ANGLE
RET (SIN :ANGLE) / COS :ANGLE
FIN

?EC TG 45
1.
```

ENT

ENT n (opération)
Retourne la partie entière de n en supprimant les décimales s'il y a lieu). Voir aussi ARRONDIS

Exemples:

```
ENT 5.2129
   retourne 5

ENT 5.5129
   retourne 5

ENT 5
   retourne 5

ENT -5.8
   retourne -5

ENT -12.3
   retourne -12
```

La procédure suivante indique si un nombre est entier ou non

```
POUR ENTP :N
SI NON NOMBREP :N [RET [PAS UN NOMBRE]]
RET :N = ENT :N
FIN

?EC ENTP 17
VRAI
?EC ENTP 10/8
FAUX
?EC ENTP "UN
PAS UN NOMBRE
?EC INTP RAC 50
FAUX
```

PRODUIT

PRODUIT a b (opération)
(PRODUIT a b c ...)
Retourne le produit des arguments a, b,... (Forme infixe équivalente *, voir ci-dessous, liste des opérations en forme infixe).

Exemples:

```
PRODUIT 6 2
   retourne 12

(PRODUIT 2 3 4)
   retourne 24

PRODUIT 2.5 4
   retourne 10.

PRODUIT 2.5 2.5
   retourne 6.25

POUR CUBE :NOMBRE
RET (PRODUIT :NOMBRE :NOMBRE :NOMBRE)
```

```
FIN

?EC CUBE 2
8
```

S'il y a un seul argument PRODUIT retourne cet argument.

QUOTIENT

QUOTIENT a b (opération)
Retourne le quotient entier de la division de a par b. Il y a erreur si b vaut 0.

Exemples:

```
QUOTIENT 12 5
   retourne 2

QUOTIENT -12 5
   retourne -2

QUOTIENT 6 2.5
   retourne 2

QUOTIENT 3.2 0
   donne une erreur
```

HASARD

HASARD n (opération)
Retourne un entier non négatif aléatoire et inférieur à n.

Exemple:

HASARD 6 pourrait retourner 0, 1, 2, 3, 4 ou 5. Le programme suivant simule le tirage d'un dé.

```
POUR DE6
RET 1 + HASARD 6
FIN

?EC DE6
3
?EC DE6
5
?EC DE6
3
```

RESTE

RESTE a b (opération)
Retourne le reste de la division entière de a par b. (Si a et b sont entiers, il s'agit de a mod b.) Il y a erreur si b est 0.

Exemples:

```
RESTE 12 10
   retourne 2
```

```
RESTE 12 5
   retourne 2

RESTE 12 15
   retourne 12

RESTE -12 5
   retourne -2
```

La procédure suivante détermine si un nombre est pair.

```
POUR PAIRP :NOMBRE
RET 0 = RESTE :NOMBRE 2
FIN

?EC PAIRP 5
FAUX
?EC PAIRP 12462
VRAI
```

La procédure suivante, plus générale, indique si un nombre est diviseur d'un autre.

```
POUR DIVISEURP :A :B
RET 0 = RESTE :B :A
FIN

?EC DIVISEURP 3 15
VRAI
?EC DIVISERP 4 15
FAUX
```

FIXEHASARD

FIXEHASARD (commande)
Rend le résultat de HASARD reproductible: après avoir effectué FIXEHASARD, les appels à HASARD retourneront chaque fois la même séquence de résultats.

Exemple:

```
POUR LANCERDE :JETS
SI :JETS = 0 [STOP]
EC 1 + HASARD 6
LANCERDE :JETS - 1
FIN

?LANCERDE 6
3
2
6
6
3
1
?LANCERDE 6
5
```

```
5
5
1
3
1
?FIXEHASARD
?LANCERDE 6
4
3
6
6
1
2
?LANCERDE 6.
4
3
6
6
1
2
```

ARRONDIS

ARRONDIS n (opération)
Retourne n arrondi au plus proche entier. Comparez avec les exemples de ENT.

Exemples:

ARRONDIS 5.2129
retourne 5

ARRONDIS 5.5129
retourne 6

ARRONDIS .5
retourne 1

ARRONDIS -5.8
retourne -6

ARRONDIS -12.3
retourne -12

SIN

SIN n (opération)
Retourne le sinus de n degrés. Il y a erreur si n est supérieur à 4.19E6 (voir COS).

Exemple:

SIN 30
retourne .5

RAC

RAC a (opération)
Retourne la racine carrée de a. Il y a erreur si a est négatif.

RAC 25
retourne 5.

RAC 259
retourne 16.0935

La procédure suivante retourne la distance entre la position de la tortue et l'origine.

```
POUR DIST.ORIGINE
RET RAC SOMME XCOR * XCOR YCOR * YCOR
FIN
```

La procédure DISTANCE prend pour arguments 2 positions et retourne la distance entre elles.

```
POUR DISTANCE :POS1 :POS2
RET RAC SOMME CARRE ((PREMIER :POS1) - PREMIER
:POS2)
CARRE ((DER :POS1) - DER :POS2)
FIN

POUR CARRE :N
RET :N * :N
FIN

?EC DISTANCE [-70 10] [50 60]
130.
```

SOMME

SOMME a b (opération)
(SOMME a b c ...)
Retourne la somme des arguments. (Forme infixe équivalente +, voir ci-dessous, liste des formes infixes).

Exemples:

```
SOMME 5 2
   retourne 7

(SOMME 13 2 -1)
   retourne 5

SOMME 2.3 2.561
   retourne 4.861
```

S'il y a un seul argument SOMME retourne cet argument.

Formes infixes

NOTE: Du fait que ces symboles d'opérations sont des

[illegible] espaces avant et [illegible] sont équivalentes:

```
2 + 5
2+5
```

+ a + b (opération forme infixe)
Retourne la somme des arguments. Équivalent à SOMME (voir ci-dessus, liste des formes préfixes).

Exemples:

```
5 + 2
   retourne 7

1 + 3 + 2 + 1
   retourne 7

2.54 + 12.3
   retourne 14.84
```

- a - b (opération forme infixe)
Retourne le résultat obtenu en soustrayant b de a. Si a est absent et s'il n'y a pas d'espace après le signe moins, l'expression retourne l'opposé de b (0 - b)

Exemples:

```
?EC 7 - 1
6
?EC 7-1
6
?EC PRODUIT 7 -1
-7
?EC -3
-3
?EC - 3
ARGUMENTS INSUFFISANTS POUR -
?EC -3 - -2
-1
```

La procédure ABS retourne la valeur absolue de l'argument.

```
POUR ABS :NOMB
RET SI :NOMB < 0 [- :NOMB] [:NOMB]
FIN

?EC ABS -35
35
?EC ABS 35
35
```

APPROXIMATION indique si 2 nombres sont "proches":

```
POUR APPROXIMATION :A :B
RET (ABS :A - :B) < .01
FIN

?EC APPROXIMATION XCOR 100
VRAI
?EC XCOR
99.9934
```

Remarquez qu'il y a une ambiguité possible entre le signe moins avec un et avec deux arguments.

```
7-1 est 6
7 - 1 est également 6
```

Mais 7 -1 est une paire de nombres (7 et -1). Pour plus de détails voir Appendice I.

*

a * b (opération forme infixe)
Retourne le produit des arguments. Equivalent à la forme préfixe PRODUIT. (Voir ci-dessus, liste des formes préfixes.)

Exemples:

```
6*2
   retourne 12

2*3*4
   retourne 24

1.3*1.3
   retourne 1.69

48*.5
   retourne 24.
```

```
POUR FACTORIELLE :N
SI :N = 0 [RET 1] [RET :N*FACTORIELLE :N-1]
FIN

?EC FACTORIELLE 4
24
?EC FACTORIELLE 1
1
```

/

a / b (opération forme infixe)
Retourne a divisé par b. Une erreur intervient si b est 0.

Exemples:

```
6/3
```

6/3
retourne 2.66667

25./3.8
retourne .657895

< a < b (opération forme infixe)
Retourne VRAI si a est inférieur à b, FAUX sinon.

Exemples:

2 < 3
retourne VRAI

-7 < -10
retourne FAUX

= a = b (opération forme infixe)
Retourne VRAI si a et b sont des nombres égaux, des mots identiques, ou des listes identiques; FAUX sinon. Equivalent à EGALP (décrit au chapitre 2).

Exemples:

100 = 50*2
retourne VRAI

3 = PREMIER "3.1416
retourne VRAI

[LE LA LES] = [LE LA]
retourne FAUX

7. = 7
retourne VRAI (un nombre avec virgule est équivalent à l'entier correspondant)

" = []
RETOURNE faux (le mot vide n'est pas la même chose que la liste vide)

> a > b (opération forme infixe)
Retourne VRAI si a est supérieur à b; FAUX sinon.

Exemples:

4 > 3
retourne VRAI

-10 > -7
retourne FAUX

Il y a plusieurs façons de définir les procédures Logo. L'une d'elles utilise la primitive POUR. Voir EDITE (chap 6) et DEFINIS (chap 12) pour les autres méthodes.

Les primitives COPIEDEF et TEXTE sont aussi en relation avec la définition des procédures. Elles sont décrites au chapitre 13.

POUR

POUR nom arg1 arg2... (commande)

```
?POUR CARRE :COTE
>AV :COTE
>DR 90
>AV :COTE
>DR 90
>AV :COTE
>DR 90
>AV :COTE
>DR 90
>FIN
CARRE EST DÉFINI
?

?POUR ANNONCE :PRENOM :NOM
>EC [NOUS AVONS LA JOIE D'ANNONCER LA NAISSANCE DE]
>EC (PH :PRENOM "Q. :NOM)
>EC [4573 GRAMMES]
>FIN
ANNONCE EST DEFINI
?
```

(Les symboles d'invite apparaissent dans ces exemples pour montrer l'effet de POUR et FIN.)

POUR indique à Logo que vous êtes en train de définir une procédure appelée nom, avec (s'il y a lieu) les arguments indiqués. (Il n'y a pas à mettre de guillemets avant nom, arg1, arg2...; POUR le fait automatiquement). L'invite change de ? à > pour vous rappeler que vous êtes en train de définir une procédure. Logo n'effectue pas les actions que vous tapez; il les interprète comme faisant partie de la définition de votre procédure. Le mot spécial FIN lui indique que vous avez terminé cette définition; vous retournez au niveau supérieur de Logo. FIN doit être tapé seul sur une ligne.

FIN

FIN (mot spécial)
FIN est nécessaire, après avoir utilisé POUR, de manière à indiquer à Logo que vous avez terminé la définition de la procédure. Il doit être seul sur la ligne.

Ce chapitre vous indique comment définir et modifier les procédures Logo en utilisant la primitive EDITE.

Logo dispose d'un éditeur de texte interactif du type page-écran. Il fournit un moyen commode de définir et modifier les procédures. Il vous est possible de définir plus d'une procédure à la fois sous l'éditeur Logo.

Si vous définissez une procédure pour la première fois, vous pouvez utiliser POUR, mais ausi EDITE.

EDITE
ED

EDITE (commande)
Edite nom(liste)
Vous place en mode éditeur Logo. S'il y a un argument donné, l'éditeur démarre avec la (ou les) définition(s) de la (ou des) procédure(s) (liste)nom dans la mémoire-tampon de l'éditeur. Si une procédure nom n'est pas définie, la mémoire-tampon de l'éditeur contient uniquement la ligne de titre: POUR nom. S'il n'y a pas d'argument, la mémoire tampon de l'éditeur a le même contenu qu'au moment où vous avez utilisé l'éditeur pour la dernière fois, à moins que vous n'ayez fait appel au graphique TORTUE ou au système de fichier, dans ce cas la mémoire tampon est vide.

CTRL-C est le moyen normal de sortir de l'éditeur. Logo lit chaque ligne de la mémoire-tampon d'édition, comme si vous l'aviez tapée hors de l'éditeur. S'il y a une définition de procédure, et si la fin de la mémoire tampon est atteinte, Logo met FIN à la définition de la procédure.

CTRL-G abandonne l'édition. Vous pouvez l'utiliser si ce que vous avez modifié ne vous satisfait pas ou bien si vous ne désirez rien modifier. Toute modification déjà apportée dans le tampon de l'éditeur sera ignorée. Si vous étiez en cours de définition, la procédure restera la même qu'avant la session d'édition. Si vous tapez CTRL-G par mégarde, vous pouvez retourner à l'éditeur par EDITE, sans avoir à préciser d'argument; vos modifications éventuelles seront cependant perdues.

Dans l'éditeur, vous avez la possibilité d'utiliser tous les moyens d'édition ci-dessous. L'appendice B donne un tableau résumé de ces moyens.

Survol de l'éditeur Logo
Lorsque vous appelez l'éditeur, l'écran change. Par exemple:

```
? EDIT "POLY

POUR POLY :COTE :ANGLE
AV :COTE
DR :ANGLE
POLY :COTE :ANGLE
FIN
```

EDITEUR LOGO

Il n'y a plus de symbole d'invite mais le curseur vous montre l'endroit où vous allez inscrire un caractère. Le texte que vous éditez se trouve dans une mémoire-tampon. Vous le voyez sur l'écran. Vous avez la possibilité de déplacer le curseur n'importe où dans le texte, pour insérer, modifier ou supprimer ce que vous voulez.

Chaque touche que vous frappez provoque une action de l'éditeur. La plupart des caractères de type machine à écrire (alphabétiques, numériques, ponctuation et RETOUR) sont directement insérés à la place indiquée par le curseur sur l'écran.

Les touches fléchées et certains caractères de contrôle ont des rôles particuliers qui vous facilitent l'édition. (Les actions de l'éditeur Logo sont basées sur les définitions d'EMACS, un éditeur largement utilisé sur les autres systèmes d'ordinateurs).

Lorsque vous tapez RETOUR, le curseur effectue un saut au début de la ligne suivante, en attente de ce que vous allez taper au clavier.

Vous disposez sur une ligne de plus de caractères que ceux qui sont visibles sur une largeur d'écran. Un point d'exclamation (!) s'affiche à l'extrême droite de la ligne (colonne 39) et le curseur se déplace à la ligne suivante. Logo fait de même hors de l'éditeur. Voici un exemple de ce qui peut apparaître sur l'écran:

```
POUR ECRMESSAGES :PERSONNE
EC PHRASE :PERSONNE [, JE VAIS TAPER U!
 N TRES LONG MESSAGE POUR VOUS]
EC PHRASE [A BIENTOT,] :PERSONNE
```

L'éditeur dispose d'une mémoire-tampon auxiliaire de ligne

[illegible] à détruire.

CTRL-K efface une ligne entière du texte et la place dans le tampon à détruire.

CTRL-Y permet de retrouver et d'inscrire cette ligne plus tard à l'endroit désigné par le curseur.

Lorsque vous sortez de l'éditeur (par CTRL-C) Logo lit chaque ligne dans la mémoire-tampon d'édition comme si vous l'aviez tapée directement.

Si les instructions contenues dans la mémoire d'édition définissent une procédure (c'est-à-dire s'il y a une ligne de titre commençant par POUR...) Logo se comporte comme si vous aviez tapé directement la définition en utilisant POUR. Si la mémoire-tampon d'édition contient la définition d'une procédure mais sans instruction FIN à la fin de cette mémoire, Logo vous aide en mettant FIN à la définition à votre place.

Si la mémoire d'édition contient des instructions Logo qui ne font pas partie d'une définition de procédure, elles seront exécutées lorsque vous sortez de l'éditeur.

Les actions d'éditions

Pour éditer des instruction Logo en dehors de l'éditeur
Hors de l'éditeur vous avez la possibilité d'intervenir sur la ligne de texte que vous êtes en train de taper (celle sur laquelle figure le curseur) en utilisant les commandes ci-dessous.

Mouvement du curseur

—>
*—> ou CTRL-F (en anglais Forward, en avant)

CTRL-F
déplace le curseur d'une position vers la droite (en avant) (CTRL-U a aussi le même effet).

CTRL-B
CTRL-B (en anglais Back) déplace le curseur d'une position vers la gauche (en arrière).

CTRL-N
CTRL-N (en anglais Next line) déplace le curseur d'une ligne à la ligne suivante. Le curseur tente de se rendre sur le caractère directement en dessous de celui sur lequel il se trouve. Si la ligne suivante est plus courte, le curseur se rend à la fin de cette dernière. Si le curseur est en fin de mémoire-tampon, il ne bouge pas.

Exemple:

```
VOICI UNE LIGNE DE TEXTE
                                    curseur sur X
ET VOICI UNE AUTRE LIGNE DE TEXTE
                                    curseur sur G
```

CETTE LIGNE CI EST SI LONGUE QU'ELLE N!
E TIENT PAS SUR L'ÉCRAN
curseur sur l'espace
VOICI LA LIGNE SUIVANTE
curseur sur T

CTRL-P — CTRL-P (en anglais Previous line) déplace le curseur d'une ligne à la ligne précédente. Il tente de se rendre à la position directement au-dessus de celle sur laquelle il se trouve. Si la ligne précédente est plus courte, le curseur se rend à la fin de cette dernière. Si le curseur est sur la première ligne de la mémoire-tampon il ne bouge pas.

CTRL-A — *CTRL-A (début de ligne) place le curseur au début de la ligne sur laquelle il se trouve.

CTRL-E — *CTRL-E (en anglais End of line) place le curseur à la fin de la ligne sur laquelle il se trouve (à la droite du dernier caractère visible de la ligne).

Dans ce qui suit, il s'agit de séquences de deux touches à frapper. Tapez d'abord ESC puis la touche indiquée.

ESC> — ESC> place le curseur à la fin de la mémoire-tampon.

ESC< — ESC< place le curseur au début de la mémoire-tampon.

Pour insérer et effacer

CTRL-M — *CTRL-M est identique à RETOUR.

CTRL-Q — *CTRL-Q insère un caractère de citation (affiché sous la forme \) dans la mémoire. Ceci nous permet de citer des caractères tels que l'espace, [,], (, etc... qui ont normalement un sens spécial pour Logo.

CTRL-O — *CTRL-O (en anglais Open line) ouvre une nouvelle ligne à la position du curseur. Ceci est équivalent à RETOUR CTRL-B.

EFFACE — La touche EFF efface le caractère immédiatement à gauche EFF du curseur. Le curseur recule d'une position. La touche <— est la touche EFF. Vous feriez bien de coller sur la face avant de cette touche une étiquette autocollante EFF. (CTRL-H a le même effet).

CTRL-D — CTRL-D efface le caractère situé à la position du curseur. Comparez avec EFFACE.

CTRL-K — CTRL-K (en anglais Kill to end of line) efface le texte à partir de la position du curseur jusqu'à la fin de la ligne sur laquelle il se trouve. Ce texte est placé dans le tampon à détruire, qui peut contenir jusqu'à 256 caractères.

CTRL-Y CTRL-Y insère à la position du curseur une copie du texte qui se trouve dans le tampon à détruire. Si vous n'êtes pas en mode d'édition, vous obtenez une copie de la dernière ligne tapée.

Pour sortir de l'éditeur

CTRL-C CTRL-C est le moyen normal de sortir de l'éditeur. En validant toutes les modifications que vous avez effectuées.

CTRL-G CTRL-G abandonne l'édition. Vous pouvez l'utiliser si ce que vous avez modifié ne vous satisfait pas ou bien si vous ne désirez rien modifier. Toute modification déjà apportée dans le tampon de l'éditeur sera ignorée. Si vous étiez en cours de définition, la procédure restera la même qu'avant la session d'édition. Si vous tapez CTRL-G par mégarde, vous pouvez retourner à l'éditeur par EDITE, sans avoir à préciser d'argument; vos modifications éventuelles seront cependant perdues.

Déroulement de l'écran

Si le texte contenu dans votre mémoire-tampon d'édition est composé d'un trop grand nombre de lignes pour figurer en entier sur l'écran, vous pouvez utiliser les commandes suivantes pour contrôler ce qui est affiché à l'écran.

Ce changement de portion de la mémoire affichée à l'écran est appelé "déroulement", parce que vous pouvez imaginer la mémoire-tampon comme un rouleau se déroulant de haut en bas ou de bas en haut à travers l'écran.

CTRL-V CTRL-V (page suivante) La "page suivante" de votre texte est amenée à l'écran. Le curseur se déplace à la dernière ligne de l'écran et celle-ci devient alors la première de la page-écran suivante. Si il n'y a pas de "page suivante" dans le texte, l'éditeur vous donne un signal sonore et ne fait rien.

ESC V ESC V (page précédente) déroule une page-écran en arrière. C'est l'action opposée de CTRL-V.

CTRL-L CTRL-L (réaffiche) déroule l'écran de telle sorte que la ligne du curseur se trouve au centre. Si la mémoire-tampon contient moins d'un demi-écran de texte, l'éditeur émet un signal sonore et ne fait rien.

Notez que CTRL-L a un sens différent lorsque vous vous trouvez en dehors de l'éditeur. Voir chapitre 10 (commandes de texte et d'écran).

EDNS EDNS (commande)
EDNS groupe
EDNS groupe liste

(forme abrégée de EDite les NomS) En l'absence d'argument, place toutes les variables et leur valeurs (excepté celles dans les groupes enfouis) dans la mémoire-tampon d'édition. EDNS avec un argument place toutes les variables du groupe indiqué dans la mémoire d'édition (même si ces groupes sont enfouis). Le tampon d'édition prend la forme suivante:

```
DONNE "VAR1 VAL1
DONNE "VAR2 VAL2
DONNE "VAR3 VAL3
.
.
.
```

On peut alors éditer ces noms de variables et leurs valeurs. Lorsque vous sortez de l'éditeur les commandes DONNE sont exécutées; par conséquent toutes les modifications que vous avez apportées aux variables ou aux valeurs en mode d'édition sont prises en compte par Logo.

Exemple:

```
?IMNS
ANIMAL EST CHAT
VITESSE EST 55
AERONEF EST [HELICOPTERE JET]
```

L'écran se présente ainsi:

```
DONNE "ANIMAL "CHAT
DONNE "VITESSE 55
DONNE "AERONEF [HELICOPTERE JET]
```

Si vous effectuez les modifications suivantes,

```
DONNE "ANIMAL "KIWI
DONNE "VITESSE 55
DONNE "AERONEF [HELICOPTERE JET BALLON]
```

puis quittez l'éditeur par CTRL-C, vous aurez alors:

```
?IMNS
ANIMAL EST KIWI
VITESSE EST 55
AERONEF EST [HELICOPTERE JET BALLON]
```

Ce chapitre décrit les moyens de contrôler l'ordre dans lequel Logo suit vos instructions. Il traite des instructions conditionnelles (si telle condition est réalisée alors fais une chose; sinon fais autre chose), des instructions d'itération (exécute une liste d'instructions une ou plusieurs fois), des instructions d'arrêt (arrête cette procédure) et des instructions de pause (interromps cette procédure en cours d'exécution mais reprends son exécution plus tard).

Voici des exemples généraux du contrôle d'exécution dans les procédures Logo:

```
POUR REPONSE :QUESTION
EC DECIDE
FIN                                (appelle DECIDE)

POUR DECIDE
SI PILOUFACE [RET "OUI]  (appelle PILOUFACE)
RET "NON
FIN

POUR PILOUFACE
RET 0 = HASARD 2
FIN

?REPONSE [VAIS-JE GAGNER À LA LOTERIE?]
OUI
```

Outre celles décrites dans ce chapitre, les primitives suivantes sont aussi en relation avec les instructions conditionnelles et de contrôle: ATTRAPE, ERREUR, VA, LABEL, RENVOIE. Elles sont expliquées au chapitre 13.

CO

CO (commande)
CO objet
(COntinue) Reprend l'exécution d'une procédure après PAUSE ou CTRL-Z; l'exécution reprend à l'endroit où elle avait été interrompue. Si CO a un argument, celui-ci devient ce que retourne PAUSE. Ceci est particulièrement utile lorsque vous avez interrompu un LISLISTE ou LISCAR par CTRL-Z, puisque l'information de CO est alors lu par LISLISTE ou LISCAR.

SI

SI pred listinstruction1 (commande ou opération)
SI pred listinstruction1 listinstruction2

Si pred est VRAI, exécute listinstruction1. Si pred est FAUX, exécute listinstruction2 (s'il y a lieu).

Dans les deux cas, si la listinstruction choisie retourne quelque chose, alors SI donne le même retour. Si la liste ne retourne rien, SI ne retourne rien.

Exemples:

La procédure DECIDE est écrite de 3 manières équivalentes. les 2 premières utilisent SI comme une commande, (l'une avec 2 arguments pour SI, l'autre avec 3) la troisième utilise SI comme une opération.

SI comme commande

```
POUR DECIDE
SI 0 = HASARD 2 [RET "OUI]
RET "NON
FIN

POUR DECIDE
SI 0 = HASARD 2 [RET "OUI] [RET "NON]
FIN
```

SI comme opération

```
POUR DECIDE
RETOURNE SI 0 = HASARD 2 ["OUI] ["NON]
FIN
```

SIFAUX

Exécute listinstruction si le résultat du plus récent TESTE était FAUX, sinon ne fait rien. (Voir TESTE).

Exemple:

```
POUR QUIZ
EC [QUELLE EST LA CAPITALE DU LIECHTEN!
STEIN?]
TESTE LISLISTE = [VADUZ]
SIVRAI [EC "CORRECT!]
SIFAUX [EC "FAUX]
FIN

?QUIZ
QUELLE EST LA CAPITALE DU LIECHTENSTEI!
N?
SAINT-QUENTIN
FAUX
```

SIVRAI

Exécute listinstruction si le résultat du plus récent TESTE était VRAI, sinon ne fait rien. (Voir TESTE).

Exemple:

```
POUR QUIZ
EC [QUI EST LE MEILLEUR?]
SI ESSAIE [MOI]
SINON [EC [JUSTE] STOP]
EC [ESSAIE AUTRE CHOSE]
QUIZ
FIN

?QUIZ
QUI EST LE MEILLEUR?
JULES
ESSAIE AUTRE CHOSE
QUI EST LE MEILLEUR?
MOI
JUSTE
```

RETOURNE

RET

RETOURNE objet (commande)
abréviation: RET

Cette instruction n'a de sens qu'à l'intérieur d'une procédure, et non au niveau supérieur. Elle fait d'objet ce que retourne votre procédure, et rend le contrôle à ce qui a appelé la procédure. Remarquez que si RETOURNE par elle-même est une instruction, la procédure qui la contient est une opération puisqu'elle retourne quelque chose. (comparez avec STOP).

Exemple:

```
POUR MOLIERE
RET [J.B. POQUELIN]
FIN

?EC PH MOLIERE [EST UN AUTEUR COMIQUE]
J.B. POQUELIN EST UN AUTEUR COMIQUE
```

QUEL retourne la position d'un élément dans une liste.

```
POUR QUEL :MEMBRE :LISTE
SI NON MEMBREP :MEMBRE :LISTE [RET 0]

SI :MEMBRE = PREMIER :LISTE [RET 1]
RET 1 + QUEL :MEMBRE SP :LISTE
FIN

?DONNE "VOYELLES [A E I O U]
?EC QUEL "E :VOYELLES
2
?EC QUEL "U :VOYELLES
5
?EC QUEL "W :VOYELLES
0
```

Voici une définition possible de l'opération valeur absolue.

```
POUR [illegible] :N
SI :N < 0 [REPT -:N] [REPT :N]
FIN
```

Pour une autre forme, voir la rubrique - (signe moins).

L'opération suivante décide si son premier argument (un caractère) est un élément constitutif de son deuxième argument (un mot).

```
POUR PRESENCE :CAR :MOT
SI VIDEP :MOT RET "FAUX
SI EGALP :CAR PREMIER :MOT RET "VRAI
RETOURNE PRESENCE :CAR SAUFPREMIER :MOT
FIN

EC PRESENCE "H "THE
VRAI
EC PRESENCE "I "THE
FAUX
EC PRESENCE "SAUF "SAUFPREMIER
FAUX
```

PAUSE

PAUSE (commande ou opération)
Cette instruction n'a de sens qu'à l'intérieur d'une procédure, non au niveau supérieur. Elle interrompt l'exécution de la procédure et vous indique que vous êtes en train d'effectuer une pause. Vous avez alors la possibilité de taper des instructions de manière interactive.

Pour vous indiquer que vous vous trouvez en PAUSE et non au niveau supérieur, l'indicateur de ligne change. Le nom de la procédure ou des procédures interrompues apparaissent suivies d'un point d'interrogation. Au cours d'une PAUSE CTRL-G n'opère pas, le seul moyen de retrouver le niveau supérieur à ce moment, est d'exécuter RENVOIE "NIVEAUSUP.

Toutes les variables locales sont accessibles pendant une PAUSE (voir EC :MAX dans l'exemple ci-dessous). L'exécution de la procédure peut être reprise en tapant CO. Si CO a un argument, alors, PAUSE est une opération, l'information de CO devient le retour de PAUSE.

Exemple:

```
POUR MARCHE :MAX
DR HASARD 360
AV HASARD :MAX
EC POS
PAUSE
MARCHE :MAX
```

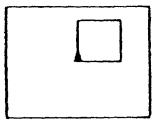
```
FIN

? MARCHE 100
60.4109 -13.947
PAUSE... DANS MARCHE:
PAUSE
MARCHE ? EC CAP
103
MARCHE ?EC :MAX
100
MARCHE ?CO
68.4381 2.1059
```

REPETE

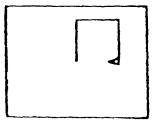
REPETE n listinstruction (commande)
Exécute listinstructions n fois. Il y a erreur si n est négatif. Si n n'est pas un entier, il est tronqué à sa partie entière.

Exemples:

REPETE 4 [AV 100 DR 90] trace un carré de 100 pas de côté.
REPETE 3 [AV 100 DR 90] trace trois quarts d'un carré.



REPETE 4
[AV 100 DR 90]



REPETE 3
[AV 100 DR 90]

EXECUTE

EXECUTE listinstruction (commande ou opération)

Exécute listinstruction comme si elle avait été tapée directement. Si listinstruction est une opération alors EXECUTE retourne ce que listinstruction retourne.

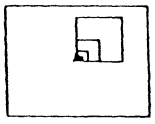
Exemples:

```
POUR CALCULATEUR
EC EXECUTE LISLISTE
EC []
CALCULATEUR
FIN

?CALCULATEUR
2 + 3
5

17.5 * 3
52.5

42 = 8 * 7
```

```
FAUX

RESTE 12 5
2

POUR TANTQUE :CONDITION :LISTINSTR
TESTE EXECUTE :CONDITION
SIFAUX [STOP]
EXECUTE :LISTINSTR
TANTQUE :CONDITION :LISTINSTR
FIN

?DR 10
?TANTQUE [XCOR < 100] [AV 25 EC POS]
```

La procédure suivante applique une commande successivement aux éléments d'une liste:

```
POUR MAP :CMDE :LISTE
SI VIDEP :LISTE [STOP]
EXECUTE LISTE :CMDE MOT "" PREMIER :LISTE
MAP :CMDE SP :LISTE
FIN

?POUR CARRE :COTE
>REPETE 4 [AV :COTE DR 90]
>FIN
?MAP "CARRE [10 20 40 80]
```



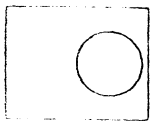
```
MAP "CARRE [10 20 40 80]

?DONNE "BENELUX [BELGIQUE PAYS.BAS LUX!
EMBOURG]
?MAP "EC :BENELUX
BELGIQUE
PAYS.BAS
LUXEMBOURG
```

La procédure suivante SANSFIN, répète son argument indéfiniment (à moins d'erreur ou d'arrêt par CTRL-G)

```
POUR SANSFIN :LISTINSTR
EXECUTE :LISTINSTR
SANSFIN :LISTINSTR
FIN
```

La commande SANSFIN [AV 1 DR 1] fait tracer un cercle à la Tortue.

```
SANSFIN [AV 1 DR 1]
```



La commande SANSFIN [EC EXECUTE LISLISTE EC []] est équivalente à la procédure CALCULATEUR définie ci-dessous.

La procédure CARRE.SUR trace un carré puis rétablit l'état du crayon à sa valeur précédente:

```
POUR CARRE.SUR
DONNE "SAUVETYPE PREMIER CRAYON
BC
CARRE 100
EXECUTE PH :SAUVETYPE
FIN

POUR CARRE :LONG
REPETE 4 [AV :LONG DR 90]
FIN

?MONTRE CRAYON
[LEVECRAYON 1]
?CARRE.SUR
?MONTRE CRAYON
[LEVECRAYON 1]
```

EXECUTE LISLISTE exécute les commandes tapées par l'usager.
EC EXECUTE LISLISTE écrit le résultat de toute expression tapée par l'usager.

STOPPE
STOP

STOP (commande)
Arrête l'exécution de la procédure courante et rend le contrôle à celle qui l'a appelée. Cette instruction n'a de sens qu'à l'intérieur d'une procédure, et non au niveau supérieur. Remarquez que la procédure contenant STOP est une commande. (Comparez à RETOURNE.)

Exemples:

```
POUR REBOURS :NUM
EC :NUM
SI :NUM = 0 [EC [C'EST PARTI] STOP]
REBOURS :NUM - 1
FIN

?REBOURS
4
3
```

```
2
1
0
C'EST PARTI
```

TESTE

TESTE pred (commande)
Se rappelle si pred est VRAI ou FAUX pour utilisation ultérieure dans SIVRAI ou SIFAUX. Chaque TESTE est local à la procédure dans laquelle il intervient.

Exemple:

```
POUR QUEST
EC [COMMENT ÇA VA?]
TESTE LISLISTE = [BIEN]
SIVRAI [EC [HEUREUX DE L'APPRENDRE]]
FIN

?QUEST
COMMENT ÇA VA?
MAL

?QUEST
COMMENT ÇA VA?
BIEN
HEUREUX DE L'APPRENDRE
```

Caractères de contrôle spéciaux

CTRL-G

Arrête immédiatement l'exécution courante, rend le contrôle au niveau supérieur de Logo et tape un point d'interrogation. Peut être tapé à n'importe quel moment.

CTRL-W

Interrompt l'exécution courante. En tapant n'importe quelle touche on peut reprendre l'exécution normale. Ceci est particulièrement utile pour vous donner le temps de lire lorsque ce qui apparaît est plus grand qu'une page-écran.

CTRL-Z

Interrompt l'exécution courante et cause une PAUSE. L'effet est équivalent à PAUSE, mais différent dans son utilisation: CTRL-Z est tapé au clavier au cours d'une exécution; PAUSE fait partie de la définition d'une procédure.

Rappelons que les "prédicats" sont des opérations qui retournent uniquement les valeurs VRAI ou FAUX. La plupart de leurs noms finissent par P (prédicat).

Il existe certains prédicats Logo dont les arguments doivent être VRAI ou FAUX. Ce sont les opérations logiques. Leurs noms ne se finissent pas par un P. Les concepteurs de Logo ont choisi de garder les noms traditionnels ET, OU et NON.

Les arguments des opérations logiques sont généralement des prédicats. On en trouvera décrits dans les autres chapitres.

Prédicats:
BOUTONP
DEFINIP
VIDEP
EGALP
TOUCHEP
LISTEP
MEMBREP
NOMP
NOMBREP
PRIMITIVEP
MONTREP
MOTP
<
=
>

ET

Retourne VRAI si tous ses arguments sont VRAI; FAUX sinon.

Exemple:

ET "VRAI "VRAI
retourne VRAI.

ET "VRAI "FAUX
retourne FAUX

ET 5 7 donne une erreur.

ET COULEUR=0 FOND=0
retourne FAUX (au lancement de la Tortue)

La procédure suivante, DECIMALP, indique si son argument est un nombre décimal.

```
POUR DECIMALP :OBJ
RETOURNE ET NOMBREP :OBJ  VIRGULEP :OBJ
FIN

POUR VIRGULEP :MOT
SI VIDEP :MOT [RET "FAUX]
SI ". = PREM :MOT [RET "VRAI]
RETOURNE VIRGULEP SAUFPREMIER :MOT
FIN

?EC DECIMALP 17
FAUX
?EC DECIMALP 17.
VRAI
?EC DECIMALP "STOP.
FAUX
```

La procédure suivante indique si la température est agréable (entre 10 et 30 degrés Celsius).

```
POUR CONFORT
SI ET :TEMPERATURE > 10 :TEMPERATURE < 30
[EC "DELECTABLE] [EC "DESAGREABLE]
FIN

?DONNE "TEMPERATURE 18
?CONFORT
DELECTABLE
```

NON

NON pred (opération)
Retourne VRAI si pred est FAUX; FAUX si pred est VRAI.

Exemples:

```
NON EGALP "A "B
   retourne Vrai

NON EGALP "A "A
   retourne FAUX

NON "A = PREMIER "CHAT
   retourne VRAI

NON "A
   retourne une erreur
```

Si MOTP n'était pas une primitive, on pourrait le définir de la manière suivante:

```
POUR MOTP :OBJ
RETOURNE NON LISTEP :OBJ
FIN
```

[illegible] si son argument est "un [illegible]".

```
POUR VRAIMOTP :OBJ
RETOURNE ET MOTP :OBJ NON NOMBREP :OBJ
FIN

?EC VRAIMOTP CAP
FAUX
?EC VRAIMOTP POS
FAUX
?EC VRAIMOTP "KANGOUROU
VRAI
?EC VRAIMOTP PREMIER CRAYON
VRAI
```

OU

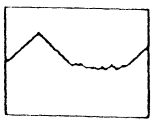
Retourne FAUX si tous ses arguments sont faux, VRAI sinon.

Exemples:

```
OU "VRAI "VRAI
   retourne VRAI

OU "VRAI "FAUX
   retourne VRAI

OU "FAUX "FAUX
   retourne FAUX

(OU "FAUX "FAUX "VRAI "FAUX)
   retourne VRAI

OU 5 7
   donne une erreur
```

La procédure MONTAGNES dessine des "montagnes".

```
POUR MONTAGNES
FIXECC 5
DR 45
AV 5
RELIEF
FIN

POUR RELIEF
AV 5 + HASARD 10
SI OU YCOR > 50 YCOR < 0
[FIXECAP 180 - CAP]
RELIEF
FIN
```



Chapitre 9 Le Monde Extérieur

Ce chapitre décrit les primitives qui permettent de communiquer avec l'ordinateur au moyen du clavier ou de périphériques spécialisés comme les manettes de jeu, et de contrôler l'affichage sur l'écran.

Outre celles décrites dans ce chapitre, les primitives CURSEUR et FIXECURSEUR concernent aussi la communication avec l'extérieur. Elles sont décrites au chapitre 10.

BOUTONP

BOUTONP numéro (opération)
Retourne VRAI si le bouton ou la manette indiquée a été pressé, FAUX sinon. (numéro doit être 0, 1 ou 2; BOUTONP 3 est une erreur, du fait que la manette no.3 n'a pas de bouton). Si rien n'est branché dans le connecteur E/S de jeu, BOUTONP retourne VRAI.

TOUCHEP

TOUCHEP (opération)
Retourne VRAI si il y a au moins un caractère en attente de lecture (c'est-à-dire, si un caractère a été tapé au clavier et pas encore saisi par LISCAR ou LISLISTE), FAUX autrement.

Exemple:

```
POUR VIRAGE
AV 2
SI TOUCHEP [TOURNE LISCAR]
VIRAGE
FIN

POUR TOURNE :DIR
SI :DIR = "D [DR 10]
SI :DIR = "G [GA 10]
FIN
```

MANETTE

MANETTE numéro (opération)
Retourne un nombre entre 0 et 255, représentant la rotation de la manette indiquée.

Exemple:

```
POUR M[illegible]E
DR (MAKE[illegible]E 0) / 25.6
AV (MAKE[illegible]E 1) / 25.6
MDESSINE
FIN
```

ECRIS
EC

ECRIS objet (commande) abréviation EC
(ECRIS objet1 objet2...)
Affiche le ou les arguments sur l'écran, suivis par un RETOUR. Les crochets extérieurs des listes ne sont pas affichés. Comparez avec TAPE ou MONTRE.

Exemples:

```
?EC "A
A
?EC "A EC [A B C]
A
ABC
?(EC "A [A B C])
A A B C
?EC []

?

POUR REECRIS :MESSAGE :COMBIEN
SI :COMBIEN < 1 [STOP]
EC :MESSAGE
EC [ ]
REECRIS :MESSAGE :COMBIEN - 1
FIN

?REECRIS [AUJOURD'HUI C'EST JEUDI] 4
AUJOURD'HUI C'EST JEUDI

AUJOURD'HUI C'EST JEUDI

AUJOURD'HUI C'EST JEUDI

AUJOURD'HUI C'EST JEUDI

?
```

LISCAR

LISCAR (opération)
Retourne le premier caractère tapé. Ce caractère peut même être un caractère de contrôle, excepté CTRL-G et CTRL-Z. Si aucun caractère n'a été lu, LISCAR attend jusqu'à ce que l'utilisateur tape quelque chose au clavier.

Exemple:
La procédure suivante, XYZZY, permet à l'utilisateur d'exécuter certains ordres en tapant une seule touche (A exécute AVANCE 5, D DROITE 10, etc. au choix). Il n'est pas nécessaire de taper RETOUR.

```
POUR XYZZY
INTERPRETE LISCAR
XYZZY
FIN

POUR INTERPRETE :CAR
SI :CAR = "A [AV 5]
SI :CAR = "D [DR 10]
FIN
```

LISLISTE

Attend que l'utilisateur tape une ligne, et retourne cette ligne sous forme de liste. Si des lignes ont déjà été tapées, LISLISTE retourne la première de celles-ci qui n'a pas encore été lue.

Exemples:

```
POUR IDENTIFIE
EC [COMMENT VOUS APPELEZ-VOUS?]
DONNE "USAGER LISLISTE
EC PH [BIENVENUE À LOGO,] :USAGER
FIN

?IDENTIFIE
COMMENT VOUS APPELEZ-VOUS?
SEBASTIEN
BIENVENUE À LOGO, SEBASTIEN
?IDENTIFIE
COMMENT VOUS APPELEZ-VOUS?
JEAN BAPTISTE MAURIN
BIENVENUE À LOGO, JEAN BAPTISTE MAURIN

POUR AGE
EC [QUEL AGE AVEZ VOUS?]
EC MESSAGE PREMIER LISLISTE
FIN

POUR MESSAGE
RET (PH [L'ANNÉE PROCHAINE VOUS AUREZ] :AGE + _ '
FIN

?AGE
QUEL AGE AVEZ-VOUS?
4
L'ANNÉE PROCHAINE VOUS AUREZ 5 ANS
QUEL AGE AVEZ-VOUS?
35
L'ANNÉE PROCHAINE VOUS AUREZ 36 ANS
```

MONTRE

Ecrit objet à l'écran, suivi d'un RETOUR. Si objet est une liste, elle est affichée avec les crochets qui l'entourent. Comparez avec TAPE et ECRIS.

Exemples:

```
?MONTRE "A
A
?MONTRE "A MONTRE [A B C]
A
[A B C]
```

TAPE

TAPE objet (commande)
(TAPE objet1 objet2...)
Affiche le ou les arguments, sans les faire suivre d'un saut de ligne (RETOUR). Les crochets entourant une liste ne sont pas affichés. Comparez avec ECRIS et MONTRE.

Exemples:

```
?TAPE "A
A? TAPE "A TAPE [A B C]
AA B C? (TAPE "A [A B C]
AA B C?
```

La procédure INVITE tape un message suivi d'un espace:

```
POUR INVITE :MESSAGE
TAPE :MESSAGE
TAPE "\                          (CTRL-Q suivi d'un espace)
FIN

POUR DEPLACER
INVITE [COMBIEN DE PAS?]
AV PREMIER LISLISTE
DEPLACER
FIN

?DEPLACER
COMBIEN DE PAS? 50
COMBIEN DE PAS? 37
COMBIEN DE PAS? 2
COMBIEN DE PAS? 108
```

ATTENDS

Ordonne à Logo d'attendre n soixantièmes de seconde.

Exemple:
La procédure RAPPORT écrit la position de la Tortue au cours de ses déplacements aléatoires. ATTENDS sert à donner à l'usager le temps de lire la position.

APPLE II dispose de 24 lignes de 40 caractères à l'écran. Vous pouvez utiliser l'écran entièrement en mode texte, ou entièrement en mode graphique. Vous pouvez aussi disposer d'un écran graphique équivalent à 20 lignes de texte, 4 lignes restant disponibles pour le texte en bas de l'écran. Au démarrage de Logo, l'écran est entièrement en mode texte.

Vous disposez de 2 moyens de modifier la destination de l'écran.

1. Des commandes Logo normales qui peuvent être tapées au niveau supérieur ou insérées au sein des procédures. (PLEINECRAN, TEXTECRAN, MIXECRAN).

2. Des caractères de contrôle particuliers qui sont lus au clavier, et ont un effet quasi-immédiat (même si une procédure est en exécution). Ces derniers caractères ne peuvent être insérés au sein d'une procédure.

En plus de celles décrites dans ce chapitre, les primitives ECHELLE et FIXECHELLE sont aussi décrites realtivement aux commandes de texte et d'écran.

VIDETEXTE

VIDETEXTE (commande)
Vide entièrement l'écran texte et place le curseur en haut et à gauche de la partie texte de l'écran. Si vous utilisez le graphique, il se placera à la quatrième ligne avant le bas de l'écran.

CURSEUR

CURSEUR (opération)
Retourne une liste de 2 nombres, les numéros de la colonne et de la ligne de la position du curseur. L'extrême gauche en haut de l'écran est [0 0] et l'extrême droite [39 0] Voir FIXECURSEUR.

Exemple: La procédure TAB passe à la position d'arrêt suivante après l'exécution d'un TAPE. Les positions d'arrêt sont toutes les huit colonnes:

```
POUR TAB
TAPE CAR 32
                                        CAR 32 est l'espace
SI (RESTE PREMIER CURSEUR 8) > 0 [TAB]
FIN

POUR TABLEAU
```

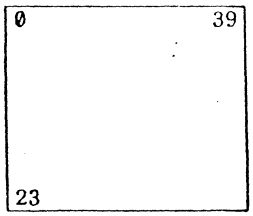
```
TAPE "[illegible] TAB TAB EC "NOTE EC []
TAPE "MARTIN TAB TAB EC 18
TAPE "EINSTEIN TAB EC 17.5
TAPE "GENEVOIS TAB TAB EC TAB19.5
```

PLEINECRAN

L'écran entier est consacré au graphique, tout ce que vous taperez restera invisible à l'écran; toutefois Logo interprétera les instructions que vous lui donnerez. Si Logo a besoin de vous renvoyer un message d'erreur, il passera de lui-même en mode MIXECRAN.

FIXECURSEUR

Place le curseur à position. Le premier élément de position est le numéro de colonne, le deuxième le numéro de ligne. Les lignes de l'écran sont numérotées de 0 à 23, les colonnes (position du caractère dans une ligne) de 0 à 39.



Il y a erreur si le numéro de ligne n'est pas compris entre 0 et 23 ou le numéro de colonne entre 0 et 39. Si position comporte un élément décimal, il est tronqué à la partie entière.

Exemple:
FIXECURSEUR [39 12] place le curseur au milieu du bord droit de l'écran.

```
POUR MVTCURS :X :Y
FIXECURSEUR LISTE (:X + PREMIER CURSEUR!
) (:Y + DERNIER CURSEUR)
FIN

?EC "A MVTCURS 2 5 EC "B
```

MIXECRAN

Partage l'écran en 2 parties: les 20 premières lignes du haut sont consacrées au graphique, les 4 du bas réservées au texte.

TEXTECRAN

Réserve l'écran entier au mode texte; le champ de la Tortue vous restera invisible jusqu'à l'exécution d'une procédure graphique.

Caractères de contrôle spéciaux

CTRL-L

Effet équivalent à PLEINECRAN. Peut être tapé n'importe quand.

Il n'a aucun effet si vous avez fait appel à l'éditeur ou

au système de fichier après votre plus récente instruction graphique, ou si vous n'avez pas du tout fait appel au graphique depuis le démarrage de Logo.

Notez que CTRL-L a un sens différent, lorsqu'il est utilisé dans l'éditeur. Voir chapitre 6.

CTRL-S — Effet équivalent à MIXECRAN. Peut être tapé n'importe quand.

Il n'a aucun effet si vous avez fait appel à l'éditeur ou au système de fichier après votre plus récente instruction graphique, ou si vous n'avez pas du tout fait appel au graphique depuis le démarrage de Logo.

CTRL-T — Effet équivalent à TEXTECRAN. Peut être tapé n'importe quand.

L'espace de travail comprend les variables, les procédures et les propriétés qui ont un sens à l'heure actuelle pour Logo. Il ne comprend pas les primitives.

Plusieurs primitives vous permettent de savoir ce que contient votre espace de travail, ainsi que d'effacer au choix variables et procédures.

Un groupe est un ensemble de procédures et de variables. De nombreuses primitives de gestion de l'espace de travail peuvent traiter un groupe comme un objet unique.

ENFOUIS

ENFOUIS groupe (commande)
Enfouit tous les noms et procédures dans un groupe (voir GROUPE). Certaines commandes (EFTOUT, EFNS, EFPS, IMTOUT, IMNS, IMPS, IMTS, SAUVE) en l'absence d'argument agissent sur tout le contenu de l'espace de travail excepté sur les groupes enfouis. L'effet d'ENFOUIS est de donner la valeur VRAI à la propriété ENFOUIS dans la liste des propriétés associée au GROUPE.

Exemple:

SAUVE "BON sauvegarde dans le fichier BON.LOGO tout l'espace de travail à l'exception des procédures et variables des groupes enfouis.

EFTOUT

EFTOUT (commande)
EFTOUT groupe
EFTOUT listegroupe
Efface de l'espace de travail toutes les procédures et variables dans groupe ou listegroupe. Voir ENFOUIS pour les exceptions.

EFFACE

EFFACE nom (commande)
EFFACE listenom
Efface la ou les procédures spécifiées de l'espace de travail.

Exemple:

EFFACE "TRIANGLE efface la procédure TRIANGLE
EFFACE [TRIANGLE CARRÉ] efface les procédures TRIANGLE et CARRÉ

EFN

EFN nom (commande)
EFN listenoms
(contraction de "EFface le Nom") Efface de l'espace de travail la ou les variables spécifiées.

Exemples:

EFN "LONG efface la variable LONG
EFN [LONG PI] efface les variables LONG et PI.

EFNS

EFNS (commande)
EFNS groupe
EFNS listegroupe
(contraction de "EFface les NomS) Efface de l'espace de travail toutes les variables de groupe ou listegroupe. Voir ENFOUIS pour les exceptions.

EFPS

EFPS (commande)
EFPS groupe
EFPS listegroupe
(contraction de EFface les ProcédureS) Efface de l'espace de travail toutes les procédures spécifiées. Voir ENFOUIS pour les exceptions.

GROUPE

GROUPE groupe nom (commande)
GROUPE groupe listenoms
Regroupe toutes les procédures spécifiées dans groupe, en donnant à la propriété PROCGRP la valeur groupe dans la liste de propriétés associée au nom de chaque procédure.

Exemples:

GROUPE "FORMES [TRIANGLE CARRE]
place TRIANGLE et CARRE dans le groupe FORMES

GRPTOUT

GRPTOUT groupe (commande)
(contraction de GRouPe TOUT) Regroupe dans groupe toutes les procédures et les variables présentes dans l'espace de travail.

IM

IM nom (commande)
IM listenoms
(contraction de IMprime) Affiche la définition de chaque procédure spécifiée.

Exemples:

```
?IM "LONG
POUR LONG :OBJ
SI VIDEP :OBJ [RET 0] [RET 1 + LONG SP!
:OBJ]
FIN

?IM [LONG SALUER]
```

```
POUR LONG :OBJ
SI VIDEP :OBJ [RET 0] [RET 1 1+LONG SP :OB!
J]
FIN

POUR SALUER
EC [BONJOUR, COMMENT CA VA?]
FIN
```

IMTOUT

IMTOUT (commande)
IMTOUT groupe
IMTOUT listegroupe
(contraction de IMprime TOUT). Affiche la définition de chaque procédure et de chaque variable (dans le ou les groupes spécifiés). Voir ENFOUIS pour les exceptions.

Exemple:

```
?IMTOUT
POUR POLY :COTE :ANGLE
AV :COTE
DR :ANGLE
POLY :COTE :ANGLE
FIN

POUR LONG :OBJ
SI VIDEP :OBJ [RET 0] [RET 1+LONG SP :OB!
J]
FIN

POUR SALUER
EC [BONJOUR, COMMENT CA VA?]
FIN

POUR SPI :COTE :ANGLE :INC
AV :COTE
DR :ANGLE
SPI :COTE + :INC :ANGLE :INC
FIN

ANIMAL EST AARDVARK
LONG EST 3.98
MONNOM EST JACQUES
```

POLY, SPI et LONGUEUR font partie du groupe FORMES.

```
?IMTOUT "FORMES
POUR POLY :COTE :ANGLE
AV :COTE
DR :ANGLE
POLY :COTE :ANGLE
FIN
```

```
POUR SPI :COTE :ANGLE :INC
AV :COTE
DR :ANGLE
SPI :COTE + :INC :ANGLE :INC
FIN

LONG EST 3.98
```

IMNS

IMNS (commande)
IMNS groupe
IMNS listegroupe
(contraction de IMprime NomS) Affiche la définition et la valeur de toutes les variables (du ou des groupes spécifiés). Voir ENFOUIS pour les exceptions.

Exemple:

```
?IMNS "LANG
F EST 3
LISTE EST [A B C]
```

IMPS

IMPS (commande)
IMPS groupe
IMPS listegroupe
(contraction de IMprime ProcédureS) Affiche la définition de chaque procédure spécifiée dans groupe ou listegroupe. Voir ENFOUIS pour les exceptions.

Exemple:

```
?IMPS
POUR POLY :COTE :ANGLE
AV :COTE
DR :ANGLE
POLY :COTE :ANGLE
FIN

POUR SPI :COTE :ANGLE :INC
AV :COTE DR :ANGLE
SPI :COTE + :INC :ANGLE :INC
FIN
```

IMTS

IMTS (commande)
IMTS groupe
IMTS listegroupe
(contraction de IMprime TitreS) Affiche la ligne de titre de chaque procédure spécifiée dans groupe ou listegroupe). Voir ENFOUIS pour les exceptions.

Exemples:

```
?IMTS
POUR POLY :COTE :ANGLE
POUR LONG :OBJ
```

```
POUR SALUT
POUR SPI :COTE :ANGLE :INC

?IMTS "FORMES
POUR POLY :COTE :ANGLE
POUR SPI :COTE :ANGLE :INC

?IMTS [FORMES LANG]
POUR POLY :COTE :ANGLE
POUR SPI :COTE :ANGLE :INC
POUR LONG :OBJ
```

DETERRE

DETERRE groupe (commande)
Déterre toutes les procédures et variables de groupe.
(Voir ENFOUIS).

Vous pouvez conserver votre espace de travail sur une disquette. L'information y est alors organisée en fichiers. C'est à vous de décider de ce que devra contenir chaque fichier.

Note: Lorsque vous utilisez une commande de fichier, l'écran graphique ou la mémoire-tampon d'édition sont effacés.

Remarquez que SAUVE ou RAMENE peuvent accepter soit un, soit deux arguments. Si vous utilisez l'une de ces instructions dans une expression, vous devrez l'entourer de parenthèses, s'il y a une autre commande sur la même ligne.

CATALOGUE

CATALOGUE (commande)
Affiche sur l'écran le nom des fichiers contenus sur la disquette.

DISQUE

DISQUE (opération)
Retourne une liste de 3 nombres: le numéro du lecteur de disque, le numéro du port sur lequel il est connecté, et le numéro de volume du disque que vous avez utilisé en dernier pour CATALOGUE ou que vous avez fixé par FIXEDISQUE. Pour plus d'information, reportez vous au Manuel APPLE DOS.

* Lorsque vous utilisez une commande de fichier l'écran graphique ou la mémoire-tampon d'édition sont effacés.

DETRUIS

DETRUIS fichier (commande)
Efface du disque le fichier nommé fichier.LOGO. Il y a erreur s'il n'est pas présent.

Exemple:

```
?DETRUIS "OURS
```
efface le fichier nommé OURS.LOGO

RAMENE

RAMENE fichier (commande)
RAMENE fichier groupe
Ramène le contenu de fichier.LOGO dans l'espace de travail comme s'il avait été tapé directement. Il y a erreur si fichier n'est pas présent. CTRL-G n'interrompt pas RAMENE.

Toutes les variables et procédures vont dans le groupe d'où elles proviennent, sauf si RAMENE a un deuxième

[illegible] ces cas, celui-ci spécifie que tout le contenu du fichier doit aller dans groupe. Si une partie du fichier est enfouie, elle le restera dans l'espace de travail.

Exemple:

?RAMENE "OURS
(tout ce qui se trouve dans le fichier OURS.LOGO passe dans l'espace de travail; chaque variable ou procédure est dans son groupe d'origine).

?RAMENE "POLYS "FORMES
(tout le contenu du fichier POLYS.LOGO passe dans le groupe FORMES dans l'espace de travail).

SAUVE

SAUVE fichier (commande)
SAUVE fichier groupe
SAUVE fichier listegroupe
Crée un fichier appelé fichier.LOGO, et y sauvegarde toutes les variables et procédures du groupe spécifié, même si elles sont enfouies, avec leurs propriétés. S'il n'y a pas de deuxième argument, tout l'espace de travail est sauvegardé, à l'exception des groupes enfouis. Si le fichier existe déjà, vous devez au préalable l'effacer au moyen de DETRUIS. SAUVE peut alors utiliser le même nom de fichier.

Les noms de fichier ne peuvent avoir plus de 30 caractères de longueur, ou ils seront tronqués à cette dimension.

FIXEDISQUE

FIXEDISQUE lecteur (commande)
FIXEDISQUE lecteur port
FIXEDISQUE lecteur port volume
Indique à Logo de fixer les numéros du lecteur disque, du port de connection et de volume aux valeurs indiquées. Le numéro de volume est un entier compris entre 1 et 254, il sert à identifier une disquette et à vous éviter d'écrire accidentellement sur la mauvaise disquette.

Ce chapitre contient une large gamme de primitives qui sont destinées à des utilisation soit rares, soit difficiles (soit les deux).

Il est divisé en 4 sections.

1. Listes de propriétés.

2. Gestion d'erreur, ATTRAPE et RENVOIE

3. Définition et redéfinition de procédures : technique avancées.

4. Primitives diverses.

Listes de propriétés

Tout mot Logo peut être associé à une liste de propriétés. Une telle liste comprend un nombre pair d'éléments. Chaque couple d'éléments contient le nom d'une propriété (par exemple COULEUR) et sa valeur (par exemple VERT).

[COULEUR VERT PATTES 1000 ANIMAL INSECTE]

Remarquez qu'une liste de propriété est donc de la forme [PROP1 VAL1 PROP2 VAL2...]. Les primitives décrites dans cette section concernent la manipulation des listes de propriétés.

En outre cinq autres primitives ont des effets directs ou indirects qui mettent en jeu des listes de propriétés ENFOUIS, DETERRE, GROUPE, GRPTOUT, RAMENE.

ENFOUIS affecte à un groupe la propriété ENFOUIS avec la valeur VRAI.

DETERRE annule la propriété ENFOUIS. Ces primitives sont décrites au chapitre 11 (gestion de l'espace de travail).

GROUPE groupe noms affecte aux procédures spécifiées la propriété PROCGRP avec la valeur groupe.

GRPTOUT groupe affecte à toutes les procédures non groupées la propriété PROCGRP avec la valeur groupe. De plus il donne à toutes les variables non groupées la propriété VALGRP avec la valeur groupe. Ces primitives sont aussi décrites au chapitre 11 (gestion de l'espace de travail).

RAMPROP groupe affecte aux procédures du fichier la propriété PROCGRP avec la valeur groupe: il affecte aussi aux noms de variables du fichier la propriété VALGRP avec la valeur groupe. Cette primitive est décrite au chapitre 12 (Les fichiers).

RPROP

RPROP nom propriété (opération)
(Signifie "Retourne PROPriété".) Retourne la valeur de la propriété du nom; retourne la liste vide si cette propriété n'existait pas.

Exemple:

```
?MONTRE RPROP "SCOLOPENDRE "PATTES
1000
[]
?MONTRE RPROP ".SYSTÈME "ENFOUIS
VRAI
```

PLISTE

PLISTE nom (opération)
Retourne la liste de propriétés associée à nom. Il s'agit d'une liste de noms de propriétés appareillées à leur valeur, sous la forme [PROP1 VAL1 PROP2 VAL2...].

Exemples:

```
?MONTRE PLISTE "SCOLOPENDRE
[COULEUR VERT PATTES 1000 ANIMAL INSECTE]
?MONTRE PLISTE ".SYSTÈME
[ENFOUIS VRAI]
```

DPROP

DPROP nom prop objet (commande)
(Signifie "DONNE PROP".) Donne à nom la propriété prop avec valeur objet. (Notez que EFTOUT, qui efface les définitions et variables de procédures, ne récupère pas l'espace qu'utilisaient les propriétés créées par DPROP. Utilisez ANNULEPROP pour effacer les propriétés.)

Exemple:

```
?MONTRE PLISTE "OISEAU
[L.D. PHENIX PATTES 2]
?DPROP "OISEAU "TAILLE [TRES GROS]
?MONTRE PLISTE "OISEAU
[TAILLE [TRES GROS] L.D. PHENIX PATTES 2]
```

EPS

EPS (commande)
EPS groupe
EPS listegroupe
(Signifie "Ecris PropriétéS".) Ecrit la ou les listes de propriétés de tout ce qui est contenu dans le ou les groupes nommés. En l'absence d'argument, écrit les listes de propriétés de tout ce que contient l'espace de travail.

Exemple:

?EPS "CARACTÈRES

IDENTITE DE L'OISEAU EST PHOENIX
COULEUR DE L'OISEAU EST JAUNE
PATTES DE L'OISEAU EST 2
TAILLE DE L'OISEAU EST (TRES GRAND)
IDENTITE DE LA GRENOUILLE EST FREDDY
COULEUR DE LA GRENOUILLE EST VERTE
PATTES DE LA GRENOUILLE EST 4

ANNULEPROP

ANNULEPROP nom prop (commande)
Retire la paire de propriété prop de la liste de propriété de nom.

Exemple:

?MONTRE PLISTE "BATEAU
[VITESSE 50 CAP 40 COULEUR VERT]
?ANNULEPROP "BATEAU "CAP
?MONTRE PLISTE "BATEAU
[VITESSE 50 COULEUR VERT]

Voir également DPROP et RPROP.

ATTRAPE

Traitement des Erreurs, ATTRAPE et RENVOIE
ATTRAPE nom listinstruction (commande)
Exécute listinstruction. S'il y a appel à RENVOIE nom pendant l'exécution de listinstruction, le contrôle revient à ATTRAPE. Le nom sert à faire correspondre un RENVOIE et un ATTRAPE. Par exemple, ATTRAPE "CHAISE [quelquechose] ratrappe un RENVOIE "CHAISE mais non un RENVOIE "TABLE.

Il y a deux cas particuliers. ATTRAPE "VRAI ratrappe n'importe quel RENVOIE; ATTRAPE "ERREUR attrape une erreur qui autrement écrirait un message d'erreur et reviendrait au niveau supérieur. Si une erreur est rattrapée, le message que Logo écrirait normalement n'est pas écrit. Voir ERREUR pour savoir comment déterminer quelle était l'erreur. Note: CTRL-G ne fonctionne pas à l'intérieur d'un ATTRAPE "ERREUR, mais RENVOIE "NIVEAUSUPÉRIEUR fonctionne.

Exemples:

1. La procédure SERPENT lit des nombres introduits par l'usager, et les considère comme des distances pour déplacer la Tortue. Elle tourne la Tortue entre les

différemment. Si l'usager tape autre chose qu'un nombre, le programme (au moyen de sa sous-procédure LISNUM) émet un message approprié et continue à fonctionner.

```
POUR SERPENT
ATTRAPPPE "NONNUM [RAMPER]
SERPENT
FIN

POUR RAMPER
EC [TAPEZ UN NOMBRE, S'IL VOUS PLAIT.]
AV LISNUM
DR 10
FIN

POUR LISNUM
LOCAL "LIGNE
DONNE "LIGNE LISLISTE
SI NON NOMBREP PREMIER :LIGNE [EC [CECI N'EST
PAS UN NOMBRE.] RENVOIE "NONNUM]
SI NOM VIDEP SP :LIGNE [EC [UN SEUL NOMBRE, S'IL
VOUS PLAIT!] RENVOIE "NONNUM]
RETOURNE PREMIER :LIGNE
FIN
```

(Notez que STOP aurait renvoyé à RAMPER, non à SERPENT)

2. La procédure FAISLE exécute des instructions introduites par l'usager. Lorsqu'il y une erreur, Logo ne tape pas le message d'erreur normal, et ne revient pas au niveau supérieur; au contraire, Logo tape [CET ÉNONCÉ EST INCORRECT] et laisse l'usager continuer à taper des instructions.

```
POUR FAISLE
ATTRAPE "ERREUR [FAISLE1]
ECRIS [CET ÉNONCÉ EST INCORRECT]
FAISLE
FIN

POUR FAISLE1
EXÉCUTE LISLISTE
FAISLE1
FIN

?FAISLE
EC 3 + 5
8
EC12 - 7
CET ÉNONCÉ EST INCORRECT
EC 12 - 7
5
RENVOIE "NIVEAUSUPÉRIEUR
?
```

ERREUR

ERREUR (opération)
Retourne une liste de six éléments contenant des informations sur l'erreur la plus récente pour laquelle il n'y a pas eu de message écrit ou retourné par ERREUR (ou la liste vide s'il n'y a pas eu d'erreur remplissant ces conditions):

. un nombre unique identifiant l'erreur (ces nombres sont indiqués en Appendice E);

. un message expliquant l'erreur;

. le nom de la procédure à l'intérieur de laquelle l'erreur s'est produite (liste vide, s'il s'agit du niveau supérieur);

. la ligne où s'est produite l'erreur;

. le nom de la primitive qui a causé l'erreur, s'il y a lieu;

. l'objet qui a causé l'erreur, s'il y a lieu.

Logo exécute RENVOIE "ERREUR chaque fois qu'il se produit une erreur, à moins que :ERRACT ne soit VRAI. Le contrôle passe au niveau supérieur, à moins que l'on est exécuté un ATTRAPE "ERREUR. Si une erreur est attrapée de cette manière, il n'y a pas écriture de message d'erreur, et vous pouvez choisir le message que vous désirez.

Si :ERRACT est VRAI, une erreur cause une pause (voir PAUSE).

Exemple:

```
POUR CARRESUR :COTE
ATTRAPE "ERREUR [REPETE 4 [AV :COTE DR 90]]
EC ERREUR
FIN

?CARRESUR "SIXPOUCES
41 [AVANCE N'AIME PAS RECEVOIR SIXPOUCES
COMME INFO] CARRESUR [ATTRAPE
"ERREUR [REPETE 4 [AVANCE :COTE DR 90]]]
AVANCE SIXPOUCES
```

RENVOIE

RENVOIE nom (commande)
Cette commande n'a de sens que dans la portée d'une commande ATTRAPE nom. Voir ATTRAPE. Il y a erreur si on ne trouve pas de ATTRAPE nom correspondant.

RENVOIE "NIVEAUSUPÉRIEUR retourne le contrôle au niveau supérieur. (Comparé à STOP.)

Définitions et redéfinitions des procédures: autres techniques

On peut représenter les procédures comme des listes, comme l'indiquent les exemples ci-dessous. Il est possible de manipuler ces représentations de liste.

COPIEDEF

COPIEDEF nouveaunom nom (commande)
Copie la définition de nom, et en fait également la définition de nouveaunom. La redéfinition ne fait partie de votre espace de travail, et ne peut être mise par SAUVE dans un fichier.

Exemples:

COPIEDEF "NOUVEAUCARRE "CARRE donne à NOUVEAUCARRE la même définition que CARRE.

COPIEDEF "A "AVANCE donne à A la même définition que AVANCE.

Ni nom ni nouveaunom ne peuvent être des noms de primitives à moins que :REDEFP ne soit VRAI. Si :REDEFP est VRAI et que nom est une primitive, alors nouveaunom est également une primitive.

DEFINIS

DEFINIS nom liste (commande)

DEFINIS "CARRE [[COTE] [REPETE 4 [AV :COTE DR 90]]]

definit la même procédure que

```
POUR CARRE :COTE
REPETE 4 [AV :COTE DR 90]
FIN
```

DEFINIS fait de liste la définition de la procédure nom. Le premier élément de liste est une liste des arguments de nom, sans les deuxpoints (:).

Si nom n'a pas d'argument, ce premier élément de liste doit être la liste vide. Chacun des éléments de la liste est une liste composée d'une ligne de la définition de la procédure. (Cette liste ne contient pas FIN car FIN ne fait pas partie de la définition de la procédure.)

Le deuxième argument de DEFINIS a la même forme que la sortie de TEXTE. Remarquez que nom ne peut être le nom d'une primitive Logo à moins que :REDEFP ne soit VRAI.

APPRENDRE est un programme qui vous permet de taper successivement les lignes d'une procédure sans argument. Chaque fois que vous pressez RETOUR, Logo exécute

l'instruction et aussi l'intégre à la définition de la procédure.

Vous pouvez effacer la ligne précédente en tapant EFFACE.

```
POUR APPRENDRE
DONNE "PRO [[]]
LISLIGNES
EC [VOULEZ VOUS GARDER CECI COMME DEFI!
NITION D'UNE PROCÉDURE?]
TESTE PREMIER PREMIER LISLISTE = "Y
SIVRAI [TAPE NOM DE LA PROCÉDURE?] DE!
FINIS PREMIER LISLISTE :PRO]
FIN

POUR LISLIGNES
DONNE "LIGNESUIV LISLISTE
SI :LIGNESUIV = [FIN] [STOP]
TESTE :LIGNESUIV = [EFFACE]
SIVRAI [ANNULE]
SIFAUX [EXECUTE :LIGNESUIV DONNE "PRO !
METF :LIGNESUIV :PRO]
LISLIGNES
FIN

POUR ANNULE
EC PH [JE VAIS EFFACER LA LIGNE] DERNIER :PRO
DONNE "PRO SD :PRO
FIN

?APPRENDRE
AV 20
DR 36
EFFACE
JE VAIS EFFACÈR LA LIGNE DR 36
DR 72
FIN
VOULEZ-VOUS GARDER CECI COMME DÉFINITI!
ON D'UNE PROCÉDURE?
OUI
NOM DE LA PROCÉDURE?PATTE

?IM "PATTE
POUR PATTE
AV 20
DR 72
FIN
```

PATTE

PRIMITIVEP

PRIMITIVEP nom (opération)
Retourne VRAI si nom est le nom d'une primitive, FAUX sinon.

Exemples:

PRIMITIVEP "AVANCE retourne VRAI
PRIMITIVEP "CARRE retourne FAUX

TEXTE

TEXTE nom (opération)
Retourne la définition de nom sous forme d'une liste de listes, adéquate comme argument de DEFINIS.

Exemple:

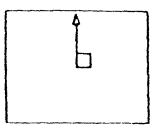
```
?MONTRE TEXTE "POLY
[[COTE ANGLE] [AV :COTE DR :ANGLE] [!
POLY :COTE :ANGLE]]
```

Le premier élément de la sortie est une liste de noms des arguments de la procédure. Les éléments restants sont des listes, dont chacune est une ligne de la définition de la procédure. (Si la procédure nom n'est pas définie, TEXTE retourne la liste vide.) L'exemple ci-dessous correspond à:

```
?IM "POLY
POUR POLY :COTE :ANGLE
AV :COTE DR :ANGLE
POLY :COTE :ANGLE
FIN
```

TEXTE peut s'utiliser en conjonction avec DEFINIS pour créer des procédures qui modifient d'autres procédures. Par exemple:

```
?IM "CARRE
POUR CARRE
REPETE 4 [AV 30 DR 30]
FIN
?DEFINIS "CARRE.AVEC.QUEUE METF [AV 100] TEXTE "CARRE

?IM "CARRE.AVEC.QUEUE
POUR CARRE.AVEC.QUEUE
REPETE 4 [AV 30 DR 90]
AV 100
FIN
```



CARRE.AVEC.QUEUE

Exemple d'utilisation:
La procédure PASAPAS modifie la définition d'une procédure pour la rendre exécutable une ligne à la fois; après l'exécution de chaque ligne, Logo attend que vous tapiez RETOUR pour continuer. NONPASAPAS rétablit la définition d'origine.

Le programme:

```
POUR PASAPAS :PRO
COPIEDEF MOT ". :PRO :PRO
DONNE "PREDEF TEXTE :PRO
DONNE "NOUVDEF (LISTE PREMIER :PREDEF)
DONNE "NOUVDEF METF (LISTE "ECRIS (LISTE
"ENTREE\"DE :PRO)) :NOUVDEF
MONTRARGS PREMIER :PREDEF
MONTRELIGNES SP :PREDEF
DEFINIS :PRO :NOUVDEF
FIN

POUR IGNORE :ENT
FIN

POUR PAS
TAPE "
IGNORE LISLISTE
FIN

POUR MONTRELIGNES :INSTRUCTIONS
SI VIDEP :INSTRUCTIONS [STOP]
DONNE "NOUVDEF METF (LISTE "TAPE PREMI!
ER :INSTRUCTIONS) :NOUVDEF
DONNE "NOUVDEF METF [PAS] :NOUVDEF
DONNE "NOUVDEF METF PREMIER :INSTRUCT!
IONS :NOUVDEF
MONTRELIGNES SP :INSTRUCTIONS
FIN

POUR MONTRARGS :LISTARG
SI VIDEP :LISTARG [STOP]
DONNE "NOUVDEF METF (LISTE "ECRIS "PHR!
ASE (LISTE (PREMIER :LISTARG) "EST (MOT ":PREMIER
:LISTARG)) :NOUVDEF
MONTRARGS SP :LISTARG
FIN

POUR NONPASAPAS
COPIEDEF :PRO MOT ". :PRO
EFFACE MOT ". :PRO
FIN
```

Utilisation du programme:

```
POUR TRIANGLE :MOT
```

```
SI VIDEP :MOT [STOP]
EC :MOT
TRIANGLE SD :MOT
FIN

?PASAPAS "TRIANGLE
?TRIANGLE "IL
ENTREE DE TRIANGLE
MOT EST IL
SI VIDEP :MOT [STOP]
                                        taper RETOUR
EC :MOT
                                        taper RETOUR
IL
TRIANGLE SD :MOT
                                        taper RETOUR
ENTREE DE TRIANGLE
MOT EST I
SI VIDEP :MOT [STOP]
                                        taper RETOUR
EC :MOT
                                        taper RETOUR
I
TRIANGLE SD :MOT
ENTREE DE TRIANGLE
MOT EST
SI VIDEP :MOT [STOP]
?
```

Primitives diverses

DEFINIP — DEFINIP mot (opération)
Retourne VRAI si mot est le nom d'une procédure, FAUX sinon.

VA — VA mot (commande)
Transfère le contrôle à l'instruction qui suit LABEL mot dans la même procédure.

Exemple:

```
POUR REBOURS :N
LABEL "BOUCLE
SI :N < 0 [STOP]
ECRIS :N
DONNE "N :N - 1
VA "BOUCLE
FIN
```

LABEL — LABEL nom (commande)
VA nom transfère le contrôle à l'instruction qui suit LABEL nom. Voir VA.

NODES — NODES (opération)

Rajoute le nombre de nodes libres. Ceci vous donne une idée de l'espace disponible dans votre espace de travail pour les procédures, les variables et l'exécution. NODES est particulièrement utile immédiatement après RECYCLE. Voir Appendice H (Espace).

RECYCLE

RECYCLE (commande)
Effectue un nettoyage, pour libérer autant de nodes que possible. (Si vous n'utilisez pas RECYCLE, il y a un nettoyage automatique lorsque c'est nécessaire, mais chacun prend au moins une seconde. L'exécution de RECYCLE avant une activité qui est fonction du temps évite l'interférence du nettoyage à un moment inapproprié). Voir NODES. Voir Appendice H (Espace).

RECOMPILE

RECOMPILE (commande)
Si vous définissez ou effacez une procédure, Logo doit parfois changer l'interprétation de plusieurs autres procédures dans l'espace de travail. Comme le nettoyage (voir RECYCLE) cette recompilation s'effectue automatiquement, donc il n'y a pas de soucis de ce côté-là. Mais comme pour le nettoyage cette activité peut ralentir un programme au mauvais moment. La commande RECOMPILE sert à effectuer immédiatement toutes les recompilations nécessaires. Après cela, il n'y a pas de risque, au moins jusqu'à la prochaine fois que vous définissez ou effacez une procédure.

.PTA

.PTA (commmande)
Entrée dans le moniteur Apple. On peut retourner à Logo par 803G suivi de RETOUR. Le nom de cette primitive comence par un point pour signaler qu'elle est dangereuse. Sauvegardez votre travail avant de l'utiliser.

.CONTENU

.CONTENU (opération)
Retourne une liste de tous les objets "connus" de Logo, c'est-à-dire vos variables et procédures, les primitives, pratiquement tout ce que vous avez tapé, et quelques autres mots. .CONTENU peut absorber beaucoup d'espace. Le nom de cette primitive commence par un point pour signaler qu'elle est dangereuse. Sauvegardez votre travail avant de l'utiliser.

.DEPOSE

.DEPOSE n a (commande)
Ecris a à l'adresse-machine n (décimal). Le nom de cette primitive commence par un point pour signaler qu'elle est dangereuse. Sauvegardez votre travail avant de l'utiliser.

.EXAMINE

.EXAMINE obj (opération)
Retourne le contenu de l'adresse-machine obj (décimal) si obj est un nombre. Sinon, retourne l'adresse du premier node d'obj. Le nom de cette primitive commence par un point pour signaler qu'elle est dangereuse. Sauvegardez votre travail avant de l'utiliser.

IMPRIMANTE

.IMPRIMANTE n (commande)
Dit à Logo que toute information normalement affichée à l'écran doit désormais être imprimée. Si n est compris entre 1 et 7 inclus, n donne le numéro du port où l'imprimante est connectée. (Avertissement: s'il n'y a pas d'imprimante au port n, Logo s'écroule). Si n est compris entre 9 et 15 inclus, n-8 donne le numéro du port, et le texte est simultanément affiché à l'écran. Quelle que soit la valeur de n, les messages d'erreur apparaissent toujours à l'écran. Le nom de cette primitive commence par un point pour signaler qu'elle est dangereuse. Sauvegardez votre travail avant de l'utiliser.

Si n a la valeur 0, la sortie revient à l'écran seul. Si n a la valeur 6, l'Apple relance Logo à partir du disque.

Exemple:

La séquence suivante imprime les définitions de toutes les procédures dans l'espace de travail sur l'imprimante connectée au port no 7, et les affiche simultanément à l'écran. Après cela, la sortie revient à l'écran seul.

```
?.IMPRIMANTE 15
?IMPS
...
?.IMPRIMANTE 0
```

Note: Les parenthèses entourant un argument indiquent que cet argument est optionnel. Un signe (#) indique une procédure qui peut recevoir un nombre quelconque d'arguments; si vous donnez à une telle procédure un nombre d'arguments différent de celui qui est indiqué, il faut entourer toute l'expression de parenthèses.

Procédures et Arguments

Graphiques Tortue
RECULE, RE distance
FOND,
NETTOIE
VIDECRAN, VE
POINT position
BARRIÈRE
AVANCE, AV distance
CAP
CACHETORTUE, CT
ORIGINE
GAUCHE, GA degrés
CRAYON
COULEURCRAYON
BAISSECRAYON, BC
GOMMECRAYON, GC
INVERSECRAYON, IC
LEVECRAYON, LC
POS
DROITE, DR degrés
ECHELLE
FIXEFOND, codecouleur
FIXECAP, degrés
FIXECRAYON, paire
FPOS, position
FIXECHELLE, n
FIXEX, FX x
FIXEY, FY y
MONTREP
MONTRETORTUE, MT
VERS, position
FENÊTRE
ENROULE
XCOR
YCOR

Mots et listes
ASCII car
SAUFPREMIER, SP objet

SAUFICHIER, SD objet
C.A.R, n
COMPTE liste
VIDEP objet
EGALP objet1 objet2
PREMIER objet
METD objet liste
ITEM n objet
DERNIER objet
LISTE objet1 objet2
LISTEP objet
METF objet liste
MEMBREP objet liste
NOMBREP objet
PHRASE, PH objet1 objet2
MOT mot1 mot2
MOTP objet
objet1 = objet2

Variables

LOCAL nom
DONNE nom objet
NOMME objet nom
NOMP mot
CHOSE nom

Opérations Arithmétiques

ARCTG n
COS degrés
ENT n
PRODUIT a b
QUOTIENT a b
HASARD n
RESTE a b
FIXEHASARD
ARRONDI n
SIN degrés
RAC a
SOMME a b
a + b
a - b
a * b
a / b
a < b
a = b
a > b

Définition et édition

EDITE, ED (nom(s))
EDNS (liste) groupe)
POUR nomproc (arguments)

Instructions conditionnelles
et contrôle d'exécution
CO (objet)
SI pred liste1 (liste2)
SIFAUX, SIF liste
SIVRAI, SIV liste
RETOURNE, RET objet
PAUSE
REPETE n liste

EXÉCUTE liste
STOP
TESTE pred

Opérations logiques
\# ET pred1 pred2
NON pred
\# OU pred1 pred2

Le Monde extérieur
BOUTONP numéro
TOUCHEP
MANETTE numéro
\# ECRIS, EC objet
LISCAR,
LISLISTE
MONTRE objet
\# TAPE objet
ATTENDS n

Commandes de texte et d'ecran
VIDETEXTE
CURSEUR
PLEINECRAN
FIXECURSEUR position
MIXECRAN
TEXTECRAN

Gestion de l'espace de travail
ENFOUIS groupe
EFTOUT (liste)groupe)
EFFACE, EF nom(s)
EFN nom(s)
EFNS (liste)groupe)
EFPS (liste)groupe)
GROUPE nom(s)
GRPTOUT groupe
IM nom(s)
IMTOUT (liste)groupe)
IMNS (liste)groupe)
IMPS (liste)groupe)
IMTS (liste)groupe)
DETERRE groupe

Fichiers
CATALOGUE
DISQUE
DETRUIS nom
RAMENE fichier (groupe)
SAUVE fichier ((liste) groupe)
FIXEDISQUE lecteur (port) (vol)

Listes de propriétés
RPROP nom prop
PLISTE nom
DPROP nom prop objet
EPS ((liste) groupe)
ANNULEPROP nom prop

Gestion d'Erreurs, ATTRAPE
et RENVOIE
ATTRAPE nom liste
ERREUR
RENVOIE nom

Définition et
redéfinition
des procédures
COPIEDEF nouveaunom nom
DEFINIS nomproc liste
PRIMITIVEP nom
TEXTE nomproc

Procédures
DEFINIP mot
VA mot
LABEL mot
NODES
RECYCLE
RECOMPILE
.PTA
.CONTENU
.DEPOSE n a
.EXAMINE n
.IMPRIMANTE port

Commandes d'Edition
*Note: une astérisque (*) indique que la commande d'édition en question fonctionne tant à l'intérieur qu'à l'extérieur de l'éditeur.
*<— ou EFFACE
*<—
*CTRL-A
*CTRL-B
CTRL-C
*CTRL-D
*CTRL-E
*CTRL-F
*CTRL-H

*CTRL-K
 CTRL-L
*CTRL-M
 CTRL-N
 CTRL-O
 CTRL-P
*CTRL-U
 CTRL-V
*CTRL-Y
 ESC V
 ESC <
 ESC >

Formules Magiques
FIN
ERRACT
ERREUR
FAUX
PROCGRP
REDEFP
STARTUP
NIVEAUSUP
VRAI
VALGRP
XYZZY
.SYSTÈME

Caractères Spéciaux
CTRL-G
CTRL-Q
CTRL-S
CTRL-T
CTRL-W
CTRL-Z

Primitives (Formes préfixes)
Note: Les parenthèses entourant un argument indiquent que cet argument est optionnel. Un signe numéro (#) indique une procédure qui peut recevoir un nombre quelconque d'arguments; si vous donnez à une telle procédure un nombre d'arguments différent de celui qui est indiqué, il faut entourer toute l'expression de parenthèses.

#	ET pred1 pred2	Retourne VRAI si tous ses arguments sont VRAI.
	ARCTG n	Retourne l'arctangente de n.
	ASCII car	Retourne le code ASCII de car.
	RECULE, RE n	Déplace la Tortue de n pas vers l'arrière.
	FOND	Retourne le numéro représentant la couleur du fond.
	ENFOUIS grp	Enfouit toutes les procédures contenues dans grp.
	SAUFPREMIER, SP objet	Retourne tout obj sauf son premier élément.
	SAUFDERNIER, SD objet	Retourne tout obj sauf son dernier élément.
	BOUTONP n	Retourne VRAI si le bouton de la manette n est enfoncée.
	CATALOGUE	Affiche les noms de tous les fichiers de la disquette.
	ATTRAPE nom liste	Exécute liste; retourne lors de l'exécution de RENVOIE nom
	CAR n	Retourne le caractère dont le code ASCII est n.
	NETTOIE	Efface l'écran graphique sans affecter la Tortue.
	VIDECRAN, VE	Efface l'écran, remet la Tortue à [0 0], avec cap 0.
	VIDETEXTE	Vide l'écran de texte.
	CO	Reprend une procédure après une pause
	COPIEDEF nouveaunom nom	Copie la définition de nom dans nouveaunom.
	COS n	Retourne le cosinus de n degrés
	COMPTE liste	Retourne le nombre d'éléments dans liste.
	CURSEUR	Retourne la position du curseur.
	DEFINIS nom liste	Donne liste comme définition de nom.
	DEFINIP	Retourne VRAI si mot est le nom d'une procédure.
	DISQUE	Retourne l'information sur le disque et le lecteur.
	POINT pos	Place un point à la position pos.
	EDITE, ED (nom(s))	Lance l'éditeur Logo (contenant les procédures nommées).
	EDNS ([liste]groupe)	Lance l'éditeur Logo (contenant les variables de [liste]groupe).
	VIDEP objet	Retourne VRAI si objet est la liste ou le

	EGALP objet1 objet2	Retourne VRAI si ses arguments sont égaux.
	EFTOUT ([liste]groupe)	Efface tout (dans [liste]groupe).
	EFFACE nom(s)	Efface les procédures nommées.
	DETRUIS nom	Détruis le fichier nom.Logo sur le disque.
	EFN nom(s)	Efface toutes les variables nommées.
	EFNS ([liste]groupe)	Efface les variables (dans [liste]groupe).
	EFPS ([liste]groupe)	Efface les procédures (dans [liste]groupe).
	ERREUR	Retourne une liste d'informations sur l'erreur la plus récente.
	BARRIÈRE	Enferme la Tortue dans la surface de l'écran
	PREMIER objet	Retourne le premier élément d'objet.
	AVANCE, AV n	Déplace la Tortue de n pas vers l'avant.
	METD objet liste	Retourne la liste formée en placant objet comme premier élément de liste.
	PLEINECRAN	Consacre tout l'écran au graphique.
	VA mot	Transfère le contrôle à LABEL mot.
	RPROP nom prop	Retourne la propriété prop de nom.
	CAP	Retourne le cap de la Tortue.
	CACHETORTUE, CT	Rend la Tortue invisible.
	ORIGINE	Place la Tortue à [0 0] et fixe le cap à 0.
	SI pred liste1 [liste2]	Si pred est VRAI, exécute liste1, sinon liste2.
	SIFAUX, SIF liste	Exécute liste si le plus récent TEST était FAUX.
	SIVRAI, SIV liste	Exécute liste si le plus récent TEST était VRAI.
	ENT n	Retourne la partie entière de n.
	ITEM n obj	Retourne le nième élément de obj.
	TOUCHEP	Retourne VRAI si une touche a été enfoncée mais pas encore lue.
	LABEL mot	Marque la ligne, pour les besoins de VA.
	DERNIER obj	Retourne le dernier élément d'obj.
	GAUCHE, GA n	Fait tourner la Tortue de n degrés vers la gauche, c'est-à-dire dans le sens antihoraire.
#	LISTE obj1 obj2	Retourne la liste formée de ces arguments.
	LISTEP obj	Retourne VRAI si obj est une liste.
	RAMENE nom (grp)	Ramène le fichier nom.Logo dans l'espace de travail (dans le groupe grp).
#	LOCAL nom	Rend la variable nom locale.
	METF obj liste	Retourne une liste formée en plaçant obj comme dernier élément de la liste.
	DONNE nom obj	Fait référer nom à obj.
	MEMBREP obj liste	Retourne VRAI si obj est un élément de liste.
	NOM obj nom	Fait d'obj la valeur de nom.
	NOMP mot	Retourne VRAI si mot réfère à un objet quelconque.
	NODES	Retourne le nombre de nodes libres.
	NON pred	Retourne VRAI si pred est FAUX.
	NOMBREP obj	Retourne VRAI si obj est un nombre.
#	OU pred1 pred2	Retourne VRAI si l'un quelconque de ses arguments est VRAI.
	RETOURNE, RET obj	Retourne le contrôle à la procédure

GROUPE groupe [obj]	Place les procédures nommées dans groupe.
MANETTE n	Retourne la valeur de la rotation du cadran de la manette n.
PAUSE	Introduit une pause dans la procédure.
CRAYON	Retourne l'état du crayon (liste contenant type et couleur).
COULEURCRAYON	Retourne un nombre représentant la couleur du crayon.
BAISSECRAYON, BC	Abaisse le crayon.
GOMMECRAYON, GC	Abaisse la gomme.
INVERSECRAYON, IC	Abaisse le crayon inverseur.
LEVECRAYON, LC	Lève le crayon.
GRPTOUT groupe	Place dans tout ce qui n'est pas déjà contenu dans un groupe.
PLISTE nom	Retourne la liste de propriétés de nom.
IM nom(s)	Ecrit les définitions des procédures nommées.
IMTOUT ([liste]groupe)	Ecrit les définitions des procédures et des noms (dans [liste]groupe).
IMNS ([liste]groupe)	Décrit les noms et valeurs des variables (dans [liste]groupe).
IMPS ([liste]groupe)	Décrit les définitions des procédures (dans [liste]groupe).
POS	Retourne la position de la Tortue.
IMTS ([liste]groupe)	Ecrit les lignes titres des procédures (dans [liste]groupe).
EPS ([liste]groupe)	Ecrit la ou les listes de propriétés de tout (dans [liste]groupe).
PRIMITIVEP nom	Retourne VRAI si nom est une primitive.
# ECRIS obj	Ecrit obj suivi de RETOUR (sans crochets pour les listes).
# PRODUIT a b	Retourne le produit de ses arguments.
QUOTIENT a b	Retourne la partie entière de a/b.
HASARD n	Retourne un entier aléatoire non-négatif inférieur à n.
LISCAR	Retourne le caractère tapé par l'usager (attend si nécessaire).
LISLISTE	Retourne la ligne tapée par l'usager (attend si nécessaire).
RECYCLE	Effectue un ramassage d'ordures.
RESTE a b	Retourne le reste de a divisé par b.
ANNULEPROP nom prop	Annule la propriété prop pour nom.
RECOMPILE	Effectue une recompilation.
REPETE n liste	Exécute liste n fois.
FIXEHASARD	Rend HASARD reproductible.
DROITE, DR n	Fait tourner la Tortue de n degrés vers la droite, c'est-à-dire dans le sens horaire.
ARRONDIS n	Retourne n arrondi à l'entier le plus proche.
EXÉCUTE liste	Exécute liste; retourne ce que liste retourne.
SAUVE nom ([liste]groupe)	Ecrit tout l'espace de travail ou, s'il y a lieu, [liste]groupe dans le fichier nom.Logo.
ECHELLE	Retourne la valeur courante d'ECHELLE.
# PHRASE, PH obj1 obj2	Retourne liste de ses arguments.
FIXEFOND n	Fixe la valeur du fond à la couleur

		représentée par n.
	FIXECURSEUR pos	Place le curseur à la position pos.
	FIXEDISQUE lecteur (port) (volume)	Fixe les numéros de lecteur, de port et de volume.
	FIXECAP n	Donne au cap de la Tortue la valeur n degrés.
	FIXECOULEUR n	Fixe la valeur de la couleur du crayon à n.
	FIXECRAYON paire	Fixe le type et la couleur du crayon aux éléments donnés par paire.
	FPOS pos	Place la Tortue à pos.
	FIXECHELLE n	Donne la valeur n au rapport d'échelle.
	FIXEX, FX x	Déplace la Tortue horizontalement jusqu'à l'abcisse x.
	FIXEY, FY y	Déplace la Tortue verticalement jusqu'à ce que l'ordonnée soit y.
	MONTRE obj	Ecrit obj suivi d'un RETOUR (avec crochets autour de liste).
	MONTREP	Retourne VRAI si la Tortue est visible.
	MONTRETORTUE, MT	Rend la Tortue visible.
	SIN n	Retourne le sinus de n degrés.
	MIXECRAN	Divise l'écran: en haut pour les graphiques en bas pour le texte.
	RAC n	Retourne la racine carrée de n.
	STOP	Arrête la procédure et retourne le contrôle à la procédure appelante.
#	SOMME a b	Retourne la somme de ses arguments.
	TESTE pred	Se rappelle si pred est VRAI ou FAUX.
	TEXTE nom	Retourne la définition de la procédure nom sous forme d'une liste.
	TEXTECRAN	Consacre tout l'écran au texte.
	CHOSE nom	Retourne l'objet désigné par nom.
	RENVOIE nom	Transfère le contrôle au ATTRAPE correspondant.
	POUR nom (arguments)	Lance la définition de la procédure nom.
	VERS pos	Retourne le cap que doit avoir la Tortue pour être orientée vers pos.
#	TAPE obj	Ecrit obj (sans crochets pour les listes et sans retour).
	DETERRE (grp)	Déterre les procédures de l'espace de travail (dans grp).
	ATTENDS n	Arrête l'exécution pendant n 60ièmes de seconde.
	FENÊTRE	Supprime les limites du champ de la Tortue.
#	MOT mot1 mot2	Retourne un mot constitué de ses arguments.
	MOTP obj	Retourne VRAI si obj est un mot.
	ENROULE	Fait enrouler le champ de la Tortue autour des bords de l'écran.
	XCOR	Retourne l'abcisse de la Tortue.
	YCOR	Retourne l'ordonnée de la Tortue.
	.PTA	Entre dans le moniteur Apple.
	.CONTENU	Retourne la liste des noms, des noms de procédures et des autres mots.
	.DÉPOSE n a	Ecrit a à l'adresse n (en notation décimale).
	.EXAMINE n	Retourne le contenu de l'adresse n (en

[illegible]

[illegible] l'écran vers l'imprimante au lieu de l'écran.

Primitives (Opérateurs infixés)

a + b	Retourne a plus b.
(a) - b	Retourne a moins b.
a * b	Retourne a fois b.
a/b	Retourne a divisé par b.
a<b	Retourne VRAI si a est inférieur à b.
obj1 = obj2	Retourne VRAI si obj1 est égal à obj2.
a>b	Retourne VRAI si a est supérieur à b.

Caractères spéciaux

Une astérisque (*) indique que la commande en question fonctionne tant à l'intérieur qu'à l'extérieur de l'éditeur.

*<— ou EFFACE	Efface les caractères à gauche du curseur.
*—>	Comme CTRL-F.
*CTRL-A	Place le curseur au début de la ligne courante.
*CTRL-B	Déplace le curseur d'un pas vers l'arrière.
*CTRL-C	Sort de l'éditeur, et lit le tampon comme s'il avait été introduit directement.
*CTRL-D	Efface le caractère à la position du curseur.
*CTRL-E	Place le curseur à la fin de la ligne courante.
*CTRL-F	Déplace le curseur d'un pas vers l'avant.
CTRL-G	Interrompt la procédure en cours et l'arrête.
*CTRL-H	Comme <—
*CTRL-K	Efface tout le contenu de la ligne courante à la droite du curseur.
CTRL-L	Déroule l'écran pour placer la ligne courante au centre (dans l'éditeur).
CTRL-L	Consacre tout l'écran aux graphiques (à l'extérieur de l'éditeur).
CTRL-M	Comme RETOUR.
CTRL-N	Déplace le curseur à la ligne suivante dans l'éditeur.
CTRL-O	Ouvre une nouvelle ligne à la position du curseur dans l'éditeur.
CTRL-P	Place le curseur sur la ligne précédente dans l'éditeur.
*CTRL-Q	Marque de citation pour le caractère suivant que vous tapez (Quote); apparaît comme une barre inversée (\).
CTRL-S	Divise l'écran: en haut pour le graphique, en bas pour le texte.
CTRL-T	Consacre tout l'écran au texte.
*CTRL-U	Comme CTRL-F.
CTRL-V	Déroule l'écran jusqu'à la page suivante dans l'éditeur.
CTRL-W	Arrête Logo jusqu'à ce qu'un autre caractère soit tapé.
*CTRL-Y	Insère le contenu du tampon à détruire à l'emplacement du curseur.
CTRL-Z	Interrompt la procédure en cours, avec une pause.

Appendice C — Formules Magiques

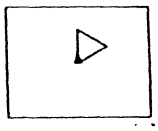
Formule	Explication
FIN	Déclare à Logo que vous avez fini de définir une procédure.
ERRACT	Variable de système: si VRAI, Logo fait une pause lorsqu'il y a erreur.
ERREUR	Marque pour RENVOIE s'il se produit une erreur.
FAUX	Argument spécial pour ET, SI, NON, OU, et TEST.
PROCGRP	Propriété d'un nom de procédure; la valeur de cette propriété est le groupe de la procédure.
REDEFP	Variable de système: si VRAI, on peut redéfinir les primitives.
STARTUP	Variable de système: si c'est une liste, Logo l'exécute après le démarrage.
NIVEAUSUP(ÉRIEUR)	Marque pour RENVOIE, permettant de retourner le contrôle au niveau supérieur.
VRAI	Argument spécial pour ET, SI, NON, OU et TÉST.
VALGRP	Propriété d'un nom de variable; sa valeur est le groupe de la variable en question.
.SYSTÈME	Groupe contenant ERRACT et REDEFP (à l'origine enfoui).

Le volume Introduction à la programmation par le graphique Tortue vous a appris à définir une procédure au moyen de l'éditeur, et à utiliser certaines commandes d'édition (CTRL-A, CTRL-B, CTRL-E, CTRL-F, CTRL-N, CTRL-P, EFFACE, CTRL-C). Il existe beaucoup de commandes d'édition, qui sont toutes décrites dans le chapitre 6 et dont la liste apparaît dans l'appendice B.

Il y a quelques autres commandes d'édition qui sont si utile que nous allons vous présenter leur utilisation sous forme d'un exemple.

Définissons la procédure TRIANGLE.

```
POUR TRIANGLE :COTE
AV :COTE
DR 120
AV :COTE
DR 120
AV :COTE
DR 120
FIN
```



TRIANGLE (ancienne version)

Imaginons que nous voulons modifier cette procédure pour qu'elle produise le dessin suivant:

TRIANGLE (nouvelle version)

Voici l'allure de la nouvelle procédure:

```
POUR TRIANGLE :COTE
AV :COTE
RE :COTE
DR 120
AV :COTE
```

```
RE :COTE
DR 120
AV :COTE
RE :COTE
DR 120
FIN
```

Nous allons EDITEr notre ancienne procédure pour la transformer en cette nouvelle procédure. Tapez

```
EDITE "TRIANGLE
```

Maintenant, au moyen de CTRL-N, placez le curseur au début de la troisième ligne

```
POUR TRIANGLE :COTE
AV :COTE
DR 120
AV :COTE
DR 120
AV :COTE
DR 120
FIN
```

Nous voulons ajouter notre nouvelle ligne à l'endroit où se trouve maintenant le curseur. On peut tout simplement taper la nouvelle instruction puis la touche RETOUR. Ceci marche très bien. Essayez de taper RE; notez que le texte de cette ligne se déplace à mesure.

```
POUR TRIANGLE :COTE
AV :COTE
REDR 120
AV :COTE
DR 120
AV :COTE
DR 120
FIN
```

Maintenant EFFACEZ les caractères que vous venez de taper (de manière que le curseur revienne au début de la ligne) et tapez CTRL-O. Cette commande de l'éditeur signifie Ouvrir nouvelle ligne. L'écran a l'allure suivante

```
POUR TRIANGLE :COTE
AV :COTE

DR 120
AV :COTE
DR 120
AV :COTE
DR 120
FIN
```

Vous avez [illegible] se trouve juste en face du curseur, puis y tapez vos nouvelles instructions. Tapez RE :COTE. Vous pouvez insérer les deux autres occurences de cette nouvelle instruction de la même manière.

```
POUR TRIANGLE :COTE
AV :COTE
RE :COTE
DR 120
AV :COTE
RE :COTE
DR 120
AV :COTE
RE :COTE
DR 120
FIN
```

Maintenant tapez CTRL-C pour sortir de l'éditeur. Essayez votre nouvelle procédure TRIANGLE.

Revenez dans l'éditeur pour apprendre quelques nouveaux trucs. Tapez EDITE "TRIANGLE.

Maintenant, placez le curseur au début de la troisième ligne en tapant CTRL-N. Tapez CTRL-K. Ceci signifie détruis (Kill) le reste de la ligne. Vous allez voir la ligne disparaître. Vous pouvez ramener la ligne en tapant CTRL-Y. Ceci signifie tire (Yank) le tampon à détruire. Essayez donc.

Lorsque vous tapez CTRL-K l'éditeur passe le texte contenu sur le reste de la ligne courante dans un emplacement particulier appelé le tampon à détruire, puis efface le texte de l'écran. Le texte reste dans le tampon à détruire. Vous pouvez alors insérer une copie de ce texte à la position du curseur (comme si vous le tapiez à nouveau) en utilisant la touche CTRL-Y. Vous pouvez utiliser cette combinaison pour déplacer des lignes dans une procédure, ou pour faire des copies d'une ligne.

Essayez la manipulation suivante:

Placez le curseur au début de la dernière ligne (comme vous voudrez, par exemple, avec CTRL-N). Tapez CTRL-O. Maintenant CTRL-Y. Ceci tire le texte du tampon à détruire. Tapez RETOUR. Tapez CTRL-Y à nouveau. Ceci place une autre copie du texte à la position du curseur.

```
POUR TRIANGLE :COTE
AV :COTE
RE :COTE
DR 120
AV :COTE
```

```
RE :COTE
DR 120
AV :COTE
RE :COTE
DR 120
RE :COTE
RE :COTE
FIN
```

Essayez diverses manières de déplacer des lignes au moyen de CTRL-O, CTRL-K, CTRL-Y. N'ayez pas peur d'abimer la procédure TRIANGLE.

Il y a une autre commande importante d'édition, qui utilise CTRL-D (en anglais DELETE character, EFFACE le caractère). Ceci efface les caractères à la position courante du curseur. Par contre, la touche EFFACE (←) efface le caractère situé à la gauche de la position courante du curseur. Ces deux commandes sont très utiles lorsque vous tapez du texte.

Amenez le curseur en divers emplacements de l'écran, et tapez CTRL-D. Voyez comment s'efface le caractère situé sous le curseur. C'est là une commande utile lorsque vous vous apercevez que vous avez fait une erreur de dactylographie. Il suffit de placer le curseur sur les caractères erronés et de les effacer.

Que se passe-t-il si vous amenez le curseur à la fin d'une ligne (avec CTRL-E) et ensuite tapez CTRL-D? Essayez. La ligne suivante "monte" et se colle à la ligne courante. Ce qui s'est passé, c'est que vous avez effacé le RETOUR à la fin de la ligne où vous trouvez. Ceci vous fait bien voir qu'il y a un RETOUR à la fin de chaque ligne, qui est un caractère réel et effaçable, même s'il est invisible sur l'écran. Remettez-le à sa place en tapant la touche RETOUR.

Maintenant, vous connaissez les éléments essentiels de l'éditeur Logo. L'allure de votre définition TRIANGLE doit être complètement chamboulée. Lorsque vous avez une telle situation, et que vous voulez revenir à Logo sans donner à Logo la nouvelle définition (chamboulée), vous pouvez sortir de l'éditeur en tapant CTRL-G. Ceci vous sort de l'éditeur sans donner d'information nouvelle à Logo. TRIANGLE n'est pas redéfini. Essayez. Vous vous retrouvez dans Logo:

```
?IM "TRIANGLE
```

Logo vous écrira la dernière définition de TRIANGLE que vous lui avez donnée:

```
POUR TRIANGLE :COTE
```

```
AV :COTE
RE :COTE
DR 120
AV :COTE
RE :COTE
DR 120
AV :COTE
RE :COTE
DR 120
FIN
```

Annexe E Messages d'erreur

Nombre	Texte
1	(procédure) EST DÉJÀ DEFINI
2	NOMBRE TROP GRAND
3	(symbole) N'EST PAS UNE PROCÉDURE
4	(symbole) N'EST PAS UN MOT
5	(procédure) NE PEUT S'UTILISER DANS UNE PROCÉDURE
6	(symbole) EST UTILISE PAR LOGO
7	NE PEUX TROUVER LE LABEL (symbole)
8	(symbole) IMPOSSIBLE EN MODE D'ÉDITION
9	(symbole) N'EST PAS DEFINI
10	(procédure) N'A PAS PRODUIT D'OBJET POUR (symbole)
11	DIFFICULTÉS DE DISQUE
12	DISQUE PLEIN
13	NE PEUX DIVISER PAR ZÉRO
14	FIN DE DONNÉES
15	FICHIER DÉJÀ EXISTANT
16	FICHIER BARRÉ
17	FICHIER INEXISTANT
18	TYPE DE FICHIER INCORRECT
19	TROP PEU D'ITEMS POUR (liste)
20	PLUS DE TAMPONS DE FICHIER
21	NE PEUX TROUVER DE TRAPPE POUR (symbole)
22	(symbole) PAS TROUVE
23	PLUS DE MEMOIRE DISPONIBLE
24	(procédure) NE PEUT S'UTILISER DANS UNE PROCÉDURE
25	(symbole) N'EST PAS VRAI OU FAUX
26	EN PAUSE...
27	VOUS ÊTES AU NIVEAU SUPÉRIEUR
28	ARRET!
29	PAS ASSEZ D'INFO POUR (procédure)
30	TROP D'ENTREE POUR (procédure)
31	TROP DANS LES PARENTHÈSES
32	TROP PEU D'ITEMS POUR (listre)
33	VALABLE DANS UNE PROCÉDURE SEULEMENT
34	TORTUE HORS LIMITES
35	JE NE SAIS PAS FAIRE (symbole)
36	(symbole) N'A PAS REÇU DE VALEUR
37	) SANS (
38	JE NE SAIS PAS QUOI FAIRE AVEC (symbole)
39	VOLUME DE DISQUE INCORRECT
40	DISQUE PROTÉGÉ POUR L'ÉCRITURE
41	(procédure) N'AIME PAS RECEVOIR (symbole) COMME ENTREE
42	(procédure) N'A PAS PRODUIT D'OBJET
	!!! ERREUR DE SYSTÈME LOGO Ne devrait pas se produire. Ecrire à SOLL

```
POUR ARCD1 :PAS :FOIS
REPETE :FOIS ADR 5 AV :PAS DR 5ü
FIN
```

```
POUR ARG1 :PAS :FOIS
REPETE :FOIS AGA 5 AV :PAS GA 5ü
FIN
```

```
POUR CERCLEG :RAYON
ARCG1 .174532 ö :RAYON 36
FIN
```

```
POUR CERCLED :RAYON
ARCD1 .174532 ö :RAYON 36
FIN
```

Commentaires: Ces procédures pour ARC et CERCLE tracent en fait un polygone à 36 côtés. (Le nombre .174532 est le résultat du calcul de 2*PI/36; PI est arrondi à 3.1416; 36 est le nombre de côtés du polygone.)

LISMOT ET LM

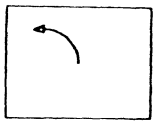
```
POUR LISMOT
RT PREMIER LISLISTE
FIN

POUR LM
RT LISLISTE
FIN
```

ARCGAUCHE
ARCG

ARCGAUCHE rayon degré (commande) abréviation: ARCG
Trace un arc du rayon donné et d'une étendue de degrés tournant vers la gauche.

Exemple:

ARCGAUCHE 30 90 trace un quart de cercle d'un rayon de 30 pas Tortue.



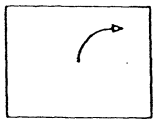
ARCG 30 90

ARCDROIT
ARCD

ARCDROIT rayon degré (commande) abréviation: ARCD
Trace un arc du rayon donnéet d'une étendue de degrés tournant vers la gauche.

Exemple:

ARCDROIT 30 90 trace un quart de cercle d'un rayon de 30 pas Tortue.



ARCD 30 90

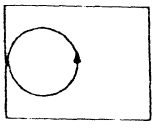
CERCLEG

CERCLEG rayon (commande)
Trace un cercle du rayon donné tournant vers la gauche.

Exemple:

CERCLEG 30 trace un cercle d'un rayon de 30 pas Tortue.

CERCLEG 30

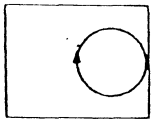


CERCLED

CERCLED rayon (commande)
Trace un cercle du rayon donné tournant vers la droite.

Exemple:

CERCLED 30 trace un cercle d'un rayon de 30 pas Tortue.



CERCLED 30

LISMOT
LM

LISMOT (opération) abréviation: **LM**
Retourne le premier mot tapé. Si le tampon d'entrée est vide, LISMOT attend jusqu'à ce que l'usager tape quelquechose. Si l'entrée est une ligne contenant plus d'un mot, LISMOT ne saisit que le premier.

Certaines des procédures définies dans le corps de ce manuel risquent de vous être utiles lorsque vous construirez vos propres procédures. Nous les avons rassemblées ici en ordre alphabétique pour plus de commodité. Consultez l'index pour retrouver les exemples de leur utilisation.

```
POUR ABS :NOMBRE
RT SI :NOMBRE < 0 Ä-:NOMBREÜ Ä:NOMBREÜ
FIN
```

```
POUR DIVISEURP :A :B
RT 0 = RESTE :B :A
FIN
```

```
POUR AJAMIS :LISTEDINSTRUCTION
EXCUTE :LISTEDINSTRUCTION
AJAMAIS :LISTEDINSTRUCTION
FIN
```

```
POUR INV.VIDEO :MOT
TYPE.SPECIAL :MOT 128
FIN
```

```
POUR TYPE.SPECIAL :MOT :ADD
SI VIDEP :MOT ÄSTOPÜ
TAPE CAR :ADD _ RESTE ASCII PREMIER :MOT 64
TYPE.SPECIAL SP :MOT :ADD
FIN
```

```
POUR ALASUITE :CMD :LISTE
SI VIDEP :LISTE ÄSTOPÜ
EXECUTE LISTE :CMD MOT "" PREMIER :LISTE
ALASUITE :CMD SP :LISTE
FIN
```

```
POUR POLY :PAS :ANGLE
AV :PAS
DR :ANGLE
POLY :PAS :ANGLE
FIN
```

```
POUR INVITE :MESSAGE
TAPE :MESSAGE
TAPE "
FIN
```

```
POUR LEQUEL :MEMBRE :LISTE
SI NON MEMBREP :MEMBRE :LISTE ÄRT 0ü
SI :MEMBRE = PREMIER :LISTE ÄRT 1ü
RETOURNE 1 _ LEQUEL :MEMBRE SP :LISTE
FIN
```

```
POUR TANTQUE :CONDITION :LISTEDINSTRUCTION
TEST EXECUTE :CONDITION
SIFAUX ÄSTOPü
EXECUTE :LISTEDINSTRUCTON
TANTQUE :CONDITION :LISTEDINSTRUCTION
FIN
```

Les procédures et variables de Logo prennent de la place; il faut encore plus d'espace pour exécuter les procédures.

Certains usagers de Logo désireront savoir combien d'espace Logo utilise, et comment l'économiser. En général, vous ne devriez pas vous occuper d'économiser de l'espace. Au lieu de cela, vous devriez vous concentrer sur la production de procédures aussi claires et aussi élégantes que possible. Toutefois, il faut reconnaître que Apple II n'a qu'une mémoire finie. Cette appendice vous présente l'allocation de l'espace dans Logo, et la manière de réduire vos besoins d'espace.

Fonctionnement

Dans Logo, l'espace est distribué en nodes, dont chacun a une longueur de 5 octets. Tous les objets et procédures Logo sont composés de nodes. Le fonctionnement interne de Logo utilise également des nodes. L'interpréteur est au courant des nodes libres, prêts à l'usage. Lorsqu'il n'y a plus de nodes libres, une section spéciale de Logo appelé ramasseur d'ordures, recherche parmi tous les nodes ceux qui ne sont pas utilisés. Par exemple, au cours de l'exécution de la procédure suivante

```
POUR ÉCRIS.SOMME :N :M
DONNE "N :N + :M
ECRIS :N
FIN
```

une fois déclaré ECRIS.SOMME 3 4, N est assigné à un nouveau node, qui contient la valeur 7. A la fin d'ECRIS.SOMME, l'ancienne valeur de N (s'il y a lieu) est rétablie, et le 7 est perdu à jamais. Le node qui contenait le 7 peut être réutilisé, et il sera récupéré comme node libre à la prochaine exécution du ramasseur d'ordures. Le ramasseur d'ordures s'exécute automatiquement si nécessaire, mais vous pouvez commander son exécution au moyen de la commande Logo RECYCLE.

L'opération NODES retourne le nombre de nodes libres; toutefois, si vous désirez vraiment savoir de combien d'espace vous disposez, il est recommandé d'utiliser l'instruction suivante:

```
?RECYCLE ECRIS NODES
259
```

Utilisation de l'espace

Chaque mot Logo utilisé n'est mis en mémoire qu'une fois: toutes les occurences de ce mot sont en fait des pointeurs vers ce mot. Un mot utilise deux nodes plus un node pour chaque groupe de deux lettres dans son nom, et un nom supplémentaire si ce mot a une liste de propriétés. Certains mots peuvent partager la fin de leur nom avec d'autres mots; par exemple, si vous déclarez MOT "SAN "FRANCISCO, le mot SAN FRANCISCO qui en résulte n'occupe que deux nodes, étant donné que la partie FRANCISCO est partagée avec le mot FRANCISCO.

Ce ne sont pas tous les mots terminés de la même manière qui partagent des lettres - principalement ceux qui sont construits avec l'opération MOT.

Un nombre, qu'il soit entier ou décimal, utilise un node. Une liste occupe un node pour chaque élément (plus la taille de l'élément lui-même). Vous pouvez avoir une idée grossière de la taille d'une procédure en déterminant la taille de la liste qui serait retournée par TEXTE.

Suggestions pour économiser l'espace

1. Important à garder à l'esprit: c'est un signe de mauvaise éducation que d'économiser de l'espace en utilisant des mots trop brefs ou obscurs, dans la rédaction des procédures, ce qui les rend moins faciles à lire.

2. Réécrivez votre programme. Remplacez les sections répétées du programme par des procédures.

3. Vous pouvez économiser de l'espace dans Logo en évitant de créer des mots nouveaux. Les noms des variables locales des procédures peuvent être les même que les noms des variables locales d'autres procédures. Les noms des procédures et des primitives peuvent également servir de noms de variables. Finalement, et dans les cas extrêmes, vous pouvez réutiliser ou même effacer les noms de primitives; ceci ne va pas détruire le mot, mais permet de récupérer deux nodes qui étaient utilisés pour stocker l'information interne sur cette primitive.

4. Il faut noter que les erreurs d'orthographe, les erreurs de dactylographie, et les mots qui ne sont plus utilisés ne sont pas détruis. Tous les mots courammant en existence apparaissent sur la liste retournée par .CONTENU. Par exemple, si vous tapez:

```
?ECRIT "HAH
JE NE SAIS PAS COMMENT ECRIT
?ABKCTUF
```

les mots ECRIT, HAH, et ABKCTUF seront créés, et ne

disparaîtront pas. Toutefois, si un mot n'a pas de valeur, de liste de propriétés ou de définition de procédure associée, il ne sera pas copié dans un fichier. Ainsi, si vous n'avez plus d'espace et qu'il y a beaucoup de ces mots (parfois appelés atomes vraiment inutiles) vous pouvez copier votre espace de travail sur un fichier, puis relancer votre Logo et ramener votre espace de travail.

Appendice I Compilation

Lorsque vous tapez une ligne pour Logo, celui-ci reconnaît les caractères comme des mots et des listes, et construit une liste qui est la représentation interne de la ligne dans Logo. Ce processus s'appelle compilation. La liste ressemble à la liste que retournerait LISLISTE. Cette appendice vous permettra de comprendre la compilation des lignes.

Délimiteurs

Un mot est d'ordinaire délimité par des espaces. Autrement dit, il y a un espace avant le mot et un autre espace avant le mot; ceci sépare le mot du reste de la ligne. Il existe quelques autres caractères délimiteurs:

[] () = <> + - * /

Il n'est pas nécessaire de taper un espace entre un mot et l'un quelconque de ces caractères. Par exemple, la ligne

```
SI 1<2[ECRIS(3+4)/5][ECRIS :X+6]
```

est compilée exactement de la même manière que

```
SI 1 > 2 [ECRIS ( 3 + 4 ) / 5 ] [ECRIS :X + 6]
```

Si vous définissez une procédure qui contient une ligne sous la première forme, vous allez voir que Logo la convertira à la seconde forme.

Pour traiter l'un quelconque des caractères mentionnés ci-dessous comme un caractère alphabétique normal, il faut le faire précéder d'une barre inclinée inverse "\" (qui est tapée au moyen de CTRL-Q). Par exemple:

```
?ECRIS "SAN\ FRANCISCO
SAN FRANCISCO                (mot unique, contenant un espace)
```

Procédures sous forme infixe

Les caractères =, <, >, +, -, *, / sont des procédures en forme infixe. On les traite comme des procédures à deux arguments, mais le nom de la procédure est écrit entre les noms des deux arguments. Il s'agit d'une propriété de la procédure, et non du nom, comme le montre l'exemple ci-dessous (pour cet exemple, il faut que REDEFP soit VRAI).

```
?COPIEDEF "PLUS "+
?ECRIS 3 PLUS 4
7
?
```

Notez: le fait qu'un symbole particulier est le nom d'une procédure infixe n'a rien à voir avec le fait qu'il s'agit d'un délimiteur: par exemple, si vous redéfinissez +, il ne sera plus une procédure infixe, mais continura à être un caractère délimiteur. (Ce genre de redéfinition n'est pas recommandé.)

Dans bien des cas (si le comportement de la procédure n'est pas ambigu), on peut utiliser les procédures infixes sous forme préfixe, comme des procédures Logo normales:

```
?ECRIS * 4 5
20
?ECRIS (+ (* 2 3) (* 4 5))
26
?ECRIS +* 2 3 * 4 5
29
```

(ce dernier exemple est compilé comme (+(*2(3*4))5)

Crochets

Le crochet ouvrant "[" et le crochet fermant "]" indiquent le début et la fin d'une liste ou sous-liste. Si on atteint la fin d'une ligne (si l'on enfonce la touche RETOUR) et qu'il reste des crochets ouverts, toutes les sous-listes sont fermées. Par exemple:

```
?REPETE 4 [ECRIS [CECI [EST [UN [TEST
CECI [EST [UN [TEST]]]
CECI [EST [UN [TEST]]]
CECI [EST [UN [TEST]]]
CECI [EST [UN [TEST]]]
```

- Si Logo trouve un crochet fermant pour lequel il n'y a pas de crochet ouvrant correspondant, le crochet fermant est ignoré. Les crochets (ainsi que les espaces) sont assez différents des autres caractères délimiteurs, car il ne s'agit pas de mots Logo.

Guillemet et deuxpoints

Il existe une exception à la règle que les caractères délimiteurs délimitent des mots. Si un mot délimiteur est précédé de guillemet ou deuxpoints, ce caractère délimiteur est compilé comme s'il s'agissait d'un mot d'une lettre sous forme "citée" ou "pointée" (alors que normalement on pourrait s'attendre que ce délimiteur délimite un mot vide sous forme citée ou pointée). Par exemple:

```
?ECRIS "+
```

```
+
?ECRIS "+ECRIS "(
+
(
?ECRIS "\+ECRIS
+ECRIS
```

(la barre inverse se tape par CTRL-Q)

Le signe moins
La compilation du signe moins, "-", est également un peu bizarre. Le problème dans ce cas est que ce même caractère sert à représenter trois choses différentes:

1. il fait partie d'un nombre pour indiquer que ce nombre est négatif, comme dans -3

2. il dénote une procédure à un argument, appelé le moins unaire, qui retourne l'inverse additif de son argument, come dans -XCOR ou -:DISTANCE

3. il dénote une procédure à deux arguments, qui retourne la différence entre son premier argument et son deuxième argument, comme dans 7-3 et XCOR-YCOR

Le compilateur essaye d'être intelligent, et de déterminer quelle est le sens voulu, selon les règles suivantes:

1. si le "-" précède immédiatement un nombre, et suit un délimiteur quelconque, sauf la parenthèse fermante ")", le nombre est interprété comme un nombre négatif. Ceci permet le comportement suivant:

ECRIS SOMME 20-20 (s'interprète comme 20 moins 20)
ECRIS 3*-4 (s'interprète comme 3 fois négatif 4)
ECRIS (3+4)-5 (s'interprète comme 3 plus 4 moins 5)
PREMIER [-3 4] (retourne -3)

2. si le "-" précède immédiatement un mot ou une parenthèse ouvrante "(", et suit un délimiteur quelconque à l'exception de la parenthèse fermante, il est interprété comme la procédure moins unaire:

```
FIXPOS LISTE :X -:Y
FIXPOS LISTE YCOR -XCOR
PREMIER[-XCOR] (retourne -)
```

3. dans tous les autres cas, "-" est interprété comme les autres caractères infixes, c'est-à-dire comme une procédure à deux arguments:
ECRIS 3-4 (s'interprète comme 3 moins 4)
ECRIS 3 - 4 (s'interprète exactement comme l'exemple précédent)
ECRIS - 3 4 (même procédure que l'exemple précédent)

Vous pouvez changer la définition d'une primitive Logo au moyen de DEFINIS. DEFINIS ne s'applique pas aux primitives, sauf si :REDEFP est VRAI. (La valeur initiale de cette variable est FAUX, pour éviter une redéfinition accidentelle des primitives).

La redéfinition de AV donnée dans l'exemple suivant vous permettrait de jouer un tour à un ami qui ne s'y attend pas.

```
DONNE "REDEFP "VRAI
DEFINIS "AV [[N] [RE :N]]
```

Essayez AV 100
Plus tard, vous pourriez restorer la définition de AV, en tapant simplement COPIEDEF "AV "AVANCE

Il y a des raisons moins superficielles de redéfinir les primitives. Par exemple, vous pouvez vouloir obtenir des renseignements sur le fonctionnement de votre programme, pour la correction. Par exemple, si vous découvrez que vos variables ont des valeurs incorrectes, il pourrait être utile d'avoir une version de DONNE qui note spécifiquement ces actions. Voici comment s'y prendre:

```
?COPIEDEF "VRAI.DONNE "DONNE
?DEFINIS "DONNE [[NOM CHOSE] [EC (PH [JE DONNE
LE NOM] :NOM "A :CHOSE) VRAI.DONNE :NOM
:CHOSE]]
```

Après avoir redéfini une primitive, il serait bon de donner la valeur FAUX à :REDEFP, pour protéger les primitives de modifications accidentelles.

Appendice K Le code ASCII

Cet appendice présente un tableau du code ASCII (en notation décimale) pour tous les caractères du Logo Apple de Systèmes Logo. Notez que les caractères peuvent être
. normaux (caractères blanc sur fond noir)
. inversés (caractères noir sur fond blanc)
. clignotants (alternance rapide entre normal et inversé)

Le Logo Apple ne peut écrire tous les caractères ASCII. Les caractères minuscules et les caractères de CTRL apparaissent comme leurs correspondants majuscules; les caractères 96, 123, 124, 125 et 126 apparaissent sous forme d'autres signes de ponctuation.

Les caractères inversés et clignotants constituent des extension du code ASCII. Il est possible qu'il ne fonctionnent pas dans d'autres réalisations de Logo.

Notes:
Pour transformer un caractère normal en caractère inversé, utilisez l'expression
CAR 128 + RESTE ASCII :CAR 64

Pour transformer un caractère normal en caractère clignotant, utilisez l'expression
CAR 192 + RESTE ASCII :CAR 64
Les lettres minuscules apparaissent sous forme de lettres majuscules sur l'écran lorsqu'on utilise Apple.

(NOTE: Le préfixe ^ signifie CTRL)

Code ASCII	Car	Code ASCII	Car	Code ASCII	Car	Code ASCII	Car
0	@	32	ESPACE	64	@	96	\
1	A	33	?	65	A	97	a
2	B	32	"	66	B	98	b
3	C	35	#	67	C	99	c
4	D	36	$	68	D	100	d
5	E	37	%	69	E	101	e
6	F	38	&	70	F	102	f
7	BELL	39	'	71	G	103	G
8	H	40	(	72	H	104	h
9	I	41	)	73	I	105	i
10	J	42	*	74	J	106	j
11	K	43	+	75	K	107	k
12	L	44	,	76	L	108	l
13	RETOUR	45	-	77	M	109	m
14	N	46	.	78	N	110	n
15	O	47	/	79	O	111	o
16	P	48	0	80	P	112	p
17	Q	49	1	81	!	113	q
18	R	50	2	82	R	114	r
19	S	51	3	83	S	115	s
20	T	52	4	84	T	116	t
21	U	53	5	85	U	117	u
22	V	54	6	86	V	118	v
23	W	55	7	87	W	119	w
24	X	56	8	88	X	120	x
25	Y	57	9	89	Y	121	y
26	Z	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	(
28	\	60	<	92	\	124	
29	]	61	=	93	]	125	)
30		62	>	94		126	
31	_	63	?	95	_	127	EFF

Code ASCII	car	Code ASCII	car	Code ASCII	car	Code ASCII	(clignotant) car
128	@	160	ESPACE	192	@	224	ESPACE
129	A	161	!	193	A	225	!
130	B	162	"	194	B	226	"
131	C	163	#	195	C	227	#
132	D	164	$	196	D	228	$
133	E	165	%	197	E	229	%
134	F	166	&	198	F	230	&
135	G	167	'	299	G	231	'
136	H	168	(	200	H	232	(
137	I	169	)	201	I	233	)
138	J	170	*	202	J	234	*
139	K	171	+	203	K	235	+
140	L	172	,	204	L	236	,
141	M	173	-	205	M	237	-
142	N	174	.	206	N	238	.
143	)	175	/	207	O	239	/
144	P	176	0	208	P	240	0
145	Q	177	1	209	Q	241	1
146	R	178	2	210	R	242	2
147	S	179	3	211	S	243	3
148	T	180	4	212	T	244	4
149	U	181	5	213	U	245	5
150	V	182	6	214	V	246	6
151	W	183	7	215	W	247	7
152	X	184	8	216	X	248	8
153	Y	185	9	217	Y	249	9
154	Z	186	:	218	Z	250	:
155	[	187	;	219	[	251	;
156	\	188	<	220	\	252	<
157	]	189	=	221	]	253	=
158		190	>	222		254	>
159	_	191	?	223	_	255	?

LISTE DES PRIMITIFS LOGO-APPLE

- .CONTENU
- .DEPOSE
- .EXAMINE
- .IMPRIMANTE
- .PTA
- .SYSTEME

ANNULEPROP
ARCTAN
ARRONDIS
ASCII
ATTENDS
ATTRAPE
AVANCE 11
BAISSECRAYON 24
BARRIERE CH1/2
BOUTONP
CACHETORTUE 24
CAP 62
CAR
CATALOGUE 20
CENTRE CH1/3
CHOSE
COMPTE
COPIEDEF
COS
COULEURCRAYON 62
CRAYON
CURSEUR
DEFINIP
DEFINIS
DEMARRAGE
DERNIER
DITERRE
DETRUIS
DISQUE
DONNE
EPROP
DROITE 12
ECHELLE 18
ECRIS 6
EDITE 14
EDNS
EFFACE 33
EFN
EFNS
EFPS
EFTOUT
EGALP
ENFOUIS
ENROULE 61
ENT
ERRACT
ERREUR
ET
EXECUTE
FAUX
FENETRE
FIN 7
FIXCAP 62
FIXCRAYON
FIXCURSEUR
FIXDISQUE
FIXECHELLE 18
FIXHASARD
FOND 62
GAUCHE 12
GOMMECRAYON CH1/5
GROUPE
GRPTOUT
HASARD 69
IM
IMNS
IMPS 32
IMTOUT
IMTS 32
INVERSECRAYON 24
IPS
ITEM
LABEL
LEVECRAYON 24
LISCAR 69
LISLISTE 69
LISTE
LISTEP
LOCAL
MANETTE
MEMBREP
METD
METF
MIXECRAN 22
MONTRE
MONTREP
MONTRETORTUE 10
MOT
MOTP
NETTOIE CH1/1
NIVEAUSUP
NODES
NOMBREP
NOMME
NOMP
NON
OU
PAUSE
PHRASE
PLEINECRAN 22
PLISTE
POINT CH1/2
POSITION 61
POUR 7
PREMIER 69
PRIMITIVEP
PROCGRP
PRODUIT
QUOTIENT
RAC
RAMENE 20
RECOMPILE
RECULE 12
RECYCLE
REDEFP
RENVOIE
REPETE 8
RESTE
RETOURNE
RPROP
SAUFDERNIER
SAUFPREMIER
SAUVE 20
SI 69
SIFAUX
SIN
SIVRAI
SOMME
STOP 67
TAPE
TEST
TEXTE
TEXTECRAN 22
TOUCHEP
VA
VALGRP
VERS
VIDECRAN 12
VIDEP
VIDETEXTE 7
VRAI
XCOR 62
YCOR 62