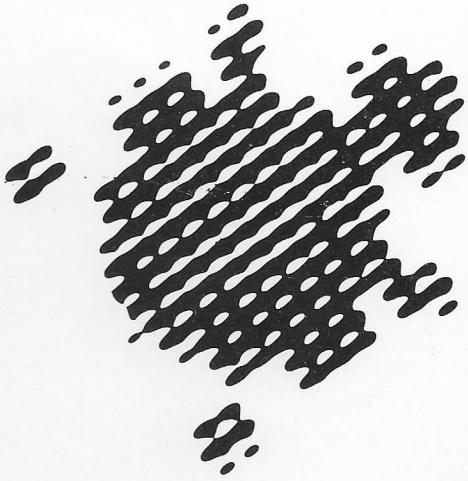
ATARI LOGO

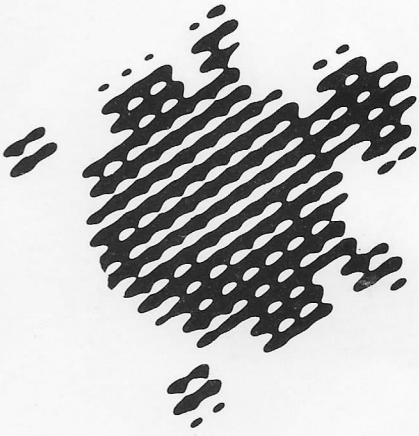
MANUEL DE REFERENCE

ATARI®

ATARI LOGO

MANUEL DE REFERENCE





ATARI®

Table des Matières

Préface

Le Manuel de référence Logo d'ATARI doit véritablement servir de référence et non de guide d'introduction pour les usagers non initiés. Ces derniers comprendront plus facilement le volume Introduction à la programmation via le graphique Tortue qui accompagne le présent manuel. Par contre, pour ceux qui ont déjà expérimenté une autre version de Logo, le Manuel de référence les aidera à se familiariser avec la version de Logo propre à ATARI.

Le Manuel de référence présente des exemples de programmes ainsi que la description des primitives Logo, commandes simples qui font partie intégrante du langage. Elles sont groupées selon une certaine organisation qu'on trouvera à la table des matières.

Ce manuel commence par deux sections très utiles. La première est une introduction à la grammaire Logo. La deuxième explique les conventions utilisées pour définir les primitives.

Il y a plusieurs façons d'employer ce manuel. L'index est utile pour repérer le numéro de la page où est décrit l'emploi d'une primitive spécifique. Les titres des chapitres de la table des matières aident à trouver la primitive convenant à une tâche particulière. Pour une référence rapide, le glossaire (appendice H), le vocabulaire Logo (appendice G) et le Guide de référence offrent un rappel suffisant.

Les appendices comprennent, entre autres, les messages qui apparaissent à l'écran, des procédures utiles, de l'information technique, les codes ASCII et un glossaire des primitives Logo d'ATARI.

Démarrage

Pour employer la cartouche du Logo d'Atari vous avez besoin d'un ordinateur domestique ATARI et d'un téléviseur ou d'un moniteur. Si vous désirez mettre vos programmes en réserve, vous devez utiliser un lecteur de disque ATARI ou un enregistreur de programme ATARI.

Si vous avez des questions spécifiques concernant le fonctionnement de votre ordinateur domestique ATARI, référez-vous au guide de l'utilisateur. Vous trouverez votre ordinateur et le Logo d'ATARI faciles à manipuler. Pour charger le Logo d'ATARI dans votre ordinateur:

1. Allumez d'abord votre téléviseur ou votre moniteur. Si vous possédez un lecteur de disque ATARI, allumez-le et attendez que le voyant lumineux s'éteigne. Si vous n'utilisez pas de lecteur de disque, passez à l'étape 3.

2. Insérez la disquette maîtresse d'ATARI dans le lecteur de disque et fermez-en la porte. Vous pouvez également utiliser une disquette fichier si elle contient les fichiers du système d'opération de disque. Dans votre matériel, vous identifierez ce disque sous l'étiquette anglaise DOS (Disk Operating System).

3. Insérez la cartouche du Logo d'ATARI dans la fente de la console destinée à cette fin et allumez votre ordinateur.

Après quelques instants vous verrez à l'écran:

```
(C) 1983 SOLI TOUS DROITS RESERVES
BIENVENUE AU LOGO D'ATARI
?
```

Le ? (point d'interrogation) est le symbole d'invite. Quand

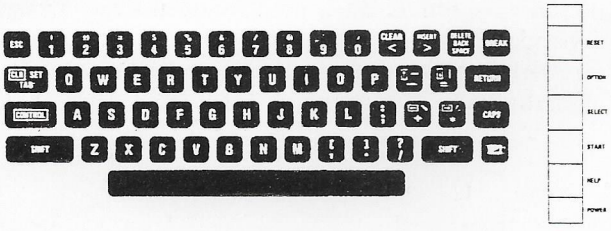
il apparaît à l'écran vous pouvez taper quelque chose. Le ■ est le curseur qui indique où s'inscrira le caractère tapé.

Le clavier

Le clavier de l'ordinateur domestique d'ATARI ressemble à celui d'une machine à écrire.

Les touches de caractères

Les touches de caractères sont A, B, C, 7, ;, $, etc. Elles comprennent les lettres de l'alphabet, les chiffres et les signes de ponctuation.



* RETOUR (RETURN)

Cette touche a une fonction de programmation. Elle dit à Logo: <<Maintenant exécute ce que je viens d'écrire>>. Pressez la touche RETOUR lorsque vous voulez que Logo obéisse à vos instructions.

Barre d'espacement (Space bar)

Cette barre imprime un espace, caractère invisible mais très important. Logo utilise les espaces comme séparateurs de mots. Par exemple Logo interpréterait CECIESTUNMOT comme un seul mot et CECI EST UN MOT comme quatre mots.

* HAUT (SHIFT)

Lorsque vous tenez cette touche enfoncée en même temps que vous pressez une touche de caractère sur laquelle sont inscrits deux symboles, c'est celui du haut que Logo affichera à l'écran. Par exemple, si vous tenez la touche HAUT enfoncée et que vous pressez la touche sur laquelle sont inscrits le point et le crochet fermant, c'est ce dernier,], que Logo affichera à l'écran.

Le crochet ouvrant, [, et le crochet fermant,], sont des symboles très importants en Logo. Ne les confondez pas avec les parenthèses, ( ), qui s'obtiennent en maintenant la touche HAUT enfoncée et en pressant en même temps la touche sur laquelle apparaissent les caractères 9 et (ou Ø et).

Pour obtenir le caractère du haut d'une touche, pressez toujours la touche HAUT pour commencer puis tenez-la enfoncée en même temps que vous pressez la touche de caractère désirée.

CTRL

Cette touche de contrôle peut changer la fonction d'une touche de caractère et transformer celle-ci en touche de commande. Si on la presse seule, rien ne se passe; mais si on frappe une autre touche en même temps, il peut se passer quelque chose. Cette combinaison n'imprime parfois rien sur l'écran mais Logo y réagit.

CTRL et les touches fléchées

CTRL <- déplace le curseur d'un espace vers la gauche et CTRL -> le déplace d'un espace vers la droite.

Les touches fléchées sont des clés d'édition utiles. Elles déplacent le curseur dans la direction où elles pointent sans affecter le texte déjà écrit. Il faut prendre note que CTRL ↑ et CTRL ↓ n'agissent qu'à l'intérieur de l'éditeur Logo d'ATARI. Une fois que le curseur est positionné, vous pouvez insérer ou effacer des caractères. Pour insérer du texte, il

s'agit simplement de placer le curseur à l'endroit voulu et de taper ce qu'on désire ajouter.

*** EFF/ARR (DELETE/BACK S)**

Cette touche efface le caractère placé à gauche du curseur, donc se déplace d'un espace en arrière.

*** ARRET (BREAK)**

La touche ARRET demande à Logo d'arrêter ce qu'il est en train d'exécuter. Elle sert également à quitter l'éditeur Logo d'ATARI sans effectuer les changements opérés. Quand vous pressez la touche ARRET, Logo imprime:

ARRET!
?

puis vous laisse taper l'instruction suivante.

*** QUITTE (ESC)**

Cette touche sert à sortir de l'éditeur Logo d'ATARI. Sa fonction et celles d'autres clés spéciales d'édition sont décrites plus en détails au chapitre 5.

ATARI (/\) ou la touche vidéo inverse (◪)

Si vous pressez la touche (/\) ou la touche (◪) puis une touche de caractère, ce dernier apparaît sur l'écran en vidéo inverse (foncé sur fond pâle). Vous retrouvez le mode normal en pressant de nouveau sur la touche (/\) ou sur la touche (◪).

*** MAJ/MIN (CAPS/LOWR)**

Lorsque vous allumez votre ordinateur domestique ATARI,

tout ce que vous tapez apparaît en majuscules. Si vous pressez d'abord la touche MAJ/MIN, tout ce que vous tapez apparaît en minuscules. Toutes les primitives Logo d'ATARI doivent être tapées en majuscules; c'est pourquoi si vous pressez accidentellement la touche MAJ/MIN, Logo ne comprendra pas vos instructions.

HAUT et MAJ/MIN combinées

Pour barrer le clavier afin qu'il n'écrive qu'en majuscules, tenez la touche HAUT enfoncée pendant que vous pressez la touche MAJ/MIN.

***REDEM (SYSTEM RESET)**

N'employez pas la touche de redémarrage du système une fois que vous avez chargé Logo car vous perdrez tout ce que contient la mémoire.

La grammaire Logo

Le langage Logo est constitué de blocs pouvant s'articuler de différentes manières selon certaines règles. Ces règles sont la grammaire du langage. Logo est donc comparable au langage humain où les règles de grammaire régissent l'ordonnance des parties du discours. Pour que Logo comprenne ce que vous en attendez, vous devez apprendre à lui donner des instructions adéquates en suivant les règles décrites dans cette section.

Procédures

Les blocs articulables de Logo sont des procédures et leurs entrées ou données. Certaines procédures sont connues de Logo car elles font partie intégrante du système Logo lui-même; c'est pourquoi elles sont appelées primitives. Vous en trouverez la liste complète à l'appendice G. Par exemple, si vous tapez:

```
VT
```

le texte disparaît de l'écran. Vous n'avez pas défini VT, mais Logo sait déjà quoi faire en réponse à cette commande.

Il existe aussi des procédures que vous définissez à l'aide des commandes POUR et EDITE. Plusieurs exemples en sont donnés dans les deux manuels du Logo d'ATARI.

Voici la définition d'une procédure:

```
POUR SALUER
ECRIS "BONJOUR
FIN
```

La première et la dernière ligne de cette définition suivent des règles spéciales. La première ligne s'appelle la ligne titre. Elle doit toujours commencer par la primitive POUR suivie du nom de la procédure. La dernière ligne ne doit

contenir que le mot FIN.

<<Définir>> une procédure et demander à Logo d'en <<exécuter>> une sont deux choses très différentes. Demander à Logo d'exécuter une procédure se dit appeler une procédure.

Par exemple, SALUER demande d'exécuter une procédure qui s'avère être la primitive ECRIS.

Il existe une autre façon de demander à Logo d'exécuter une procédure. Il s'agit de taper le nom de cette dernière quand Logo est au niveau supérieur. Ceci se reconnaît par le point d'interrogation qui apparaît à la gauche de l'écran. Un exemple en a déjà été montré avec VT.

En voici un autre:

```
SALUER
BONJOUR
```

Si vous tapez un mot dont Logo ne peut trouver la définition, un message d'erreur est affiché. Par exemple, si vous avez l'intention de créer ROUTE mais que vous en tapez le nom avant d'en définir la procédure, vous obtiendrez:

```
ROUTE
ROUTE NON DEFINIE
```

A l'intérieur de la définition d'une procédure, vous pouvez évidemment appeler une procédure que vous avez définie.

```
POUR VA.ET.VIENT
SALUER
ECRIS "AUREVOIR
FIN

VA.ET.VIENT
BONJOUR
AUREVOIR
```

SALUER est une sous-procédure de VA.ET.VIENT alors que VA.ET.VIENT est une superprocédure de SALUER.

Les données d'une procédure

Certaines procédures ont besoin de données. Par exemple:

```
ECRIS "BONJOUR
BONJOUR
```

Le mot "BONJOUR est une donnée de ECRIS. Le guillemet, ", demande à Logo de considérer le mot BONJOUR en lui-même, et non comme le nom d'une autre procédure. Voici ce qui arrive si vous omettez la donnée:

```
ECRIS
MANQUE D'ENTREES POUR ECRIS
```

Vous pouvez utiliser une phrase au lieu d'un mot comme donnée de ECRIS. Il faut alors placer l'expression entre crochets.

```
ECRIS [PASSEZ UNE BONNE JOURNEE]
PASSEZ UNE BONNE JOURNEE
```

Les procédures que vous définissez peuvent aussi avoir des données. Quand une procédure que vous avez définie s'exécute, ses données sont contenues dans des variables. Une variable est comparable à un contenant qui a un nom et qui peut recevoir un objet. Ce dernier peut être un mot ou une liste comme dans les exemples des données de ECRIS citées plus haut. Lorque vous définissez une procédure avec données, celles-ci doivent avoir chacune leur contenant ou variable. Le nom de chaque variable, précédé de deux points, s'écrit sur la ligne titre après le nom de la procédure. Par exemple:

```
POUR ACCUEILLIR :NOM
EC "BONJOUR
EC :NOM
EC [PASSEZ UNE BONNE JOURNEE]
```

```
FIN
```

La ligne titre indique à Logo que la procédure ACCUEILLIR a une seule donnée baptisée NOM. Le corps de la procédure appelle trois fois la procédure ECRIS, dont EC est l'abréviation. Le deuxième appel d'ECRIS utilise l'entrée NOM. Voici un exemple où est demandée l'exécution de ACCUEILLIR au niveau supérieur.

```
ACCUEILLIR "JEANNE
BONJOUR
JEANNE
PASSEZ UNE BONNE JOURNEE
```

Ici, la donnée de ACCUEILLIR est JEANNE. Logo en fait la valeur de NOM quand il exécute la procédure. Ainsi, dans le cas présent, ECRIS :NOM fait la même chose que ECRIS "JEANNE.

Le guillemet, les deux points et les crochets

Lorsque vous demandez à Logo d'exécuter une procédure, vous devez être très attentif à la façon dont vous écrivez les données. Ayez toujours en tête que Logo interprète chaque mot comme une demande d'exécution de procédure à moins que vous ne lui indiquiez clairement que tel n'est pas le cas. Par exemple:

```
ACCUEILLIR JEANNE
JEANNE NON DEFINIE
```

Logo interprète JEANNE comme une procédure, mais comme il ne peut en trouver la définition, il ne sait comment l'exécuter. Voici un exemple où Logo trouve une définition:

```
ACCUEILLIR SOMME 31 28
BONJOUR
59
PASSEZ UNE BONNE JOURNEE
```

SOMME est une procédure qui additionne ses données. C'est

une primitive; c'est pourquoi Logo sait comment l'exécuter même si vous ne l'avez pas définie. Au chapitre suivant, nous expliquerons davantage l'utilisation de procédures comme données.

Certains caractères sont utilisés de manière spéciale pour avertir Logo qu'une donnée n'est pas une demande d'exécution de procédure.

Un mot commençant par un guillemet, par exemple "BONJOUR, avertit Logo que cette donnée est le mot lui-même et rien d'autre. Ceci est un mot littéral. Les chiffres sont considérés comme des mots littéraux mais n'ont pas besoin d'être cités.

Le deux points devant un mot, par exemple :NOM, informe Logo qu'il s'agit du nom d'une variable et que la donnée en est la valeur.

Une séquence de mots placés entre crochets, par exemple [PASSEZ UNE BONNE JOURNEE], indique que la donnée est une liste.

L'utilisation de ces trois caractères spéciaux est démontrée dans la définition de ACCUEILLIR.

EC "BONJOUR demande à Logo d'afficher le mot BONJOUR.

EC :NOM avertit Logo d'afficher la valeur de NOM lorsqu'il exécute la procédure.

EC [PASSEZ UNE BONNE JOURNEE] indique que Logo doit afficher la liste PASSEZ UNE BONNE JOURNEE. Remarquez que Logo n'affiche pas les crochets. Si vous voulez qu'ils le soient, utilisez la commande ECRISC (ou ECC) au lieu de EC.

La différence entre commandes et opérations

Il y a deux sortes de procédures en Logo. Celles qui donnent une valeur, comme SOMME, sont appelées opérations. Celles

qui n'en donnent pas, comme ECRIS, s'appellent commandes. La distinction entre ces deux types de procédures est très importante. C'est pourquoi on indique, à chaque fois qu'une procédure est mentionnée, s'il s'agit d'une commande ou d'une opération.

L'une des principales raisons de cette distinction est le fait qu'une opération ne peut être autre chose que la donnée d'une procédure. En conséquence, chaque ligne Logo doit obligatoirement commencer par une commande. Un exemple de ceci a déjà été vu dans SOMME 31 28. Voici d'autres exemples:

```
ECRIS HASARD 2
1
```

Ce que donne HASARD 2 est la donnée de ECRIS. La donnée de HASARD est 2. Quand HASARD 2 est exécuté, le résultat en est donné à ECRIS.

```
ECRIS SOMME 3 2
5
```

Le résultat du calcul des données 3 et 2 de la procédure SOMME est donné à ECRIS.

```
ECRIS SOMME 3 PRODUIT 5 2
13
```

Ce que donne PRODUIT est la deuxième donnée de SOMME.

Si vous essayez d'utiliser une commande comme donnée, voici ce qui arrive:

```
ECRIS AVANCE 25
AVANCE N'A RIEN DONNE A ECRIS
```

Vous obtenez ce message d'erreur parce que AVANCE est une commande. Jusqu'à maintenant nous n'avons considéré que les primitives Logo. Cependant, toutes les primitives que vous définissez sont, elles aussi, soit des commandes, soit des opérations. Par exemple, la procédure ACCUEILLE

définie précédemment est une commande. La procédure suivante, TIRAGE, est une opération.

```
POUR TIRAGE
SI (HASARD 2) = Ø [RT "PILE] [RT "FACE]
FIN
```

Cette procédure donne le mot PILE si HASARD 2 donne Ø, ou le mot FACE si HASARD 2 donne 1. Tout comme dans le cas des primitives, si le nom seul de la procédure est tapé, il y a, message d'erreur.

```
TIRAGE
NE SAIS QUE FAIRE DE PILE
```

ou

```
NE SAIS QUE FAIRE DE FACE
```

Voici ce que vous obtiendrez en tapant:

```
ECRIS TIRAGE
PILE
```

ou

```
FACE
```

Presque toutes les procédures du manuel Introduction à la programmation via le graphique Tortue sont des commandes. Cependant, les procédures utilisant des mots, des listes et des nombres sont très souvent des opérations. Il faut toujours utiliser la commande RETOURNE (ou RT) pour bâtir des opérations. Pour plus de détails consultez le chapitre 6 sous la rubrique RETOURNE.

Les variables

Tel que mentionné préalablement, les variables en Logo sont comparables à des contenants portant un nom et pouvant recevoir différentes choses comme contenu. Les deux points

qui précèdent un mot avertissent Logo de rendre le contenu accessible à la procédure. Si vous tapez:

```
ECRIS :JEAN
```

Logo cherche un contenant nommé JEAN. S'il en trouve un, il donne ce qu'il contient à ECRIS. La primitive ECRIS affiche alors à l'écran le contenu, ou valeur, de JEAN. Si ECRIS ne trouve rien, Logo affiche un message d'erreur:

```
JEAN N'EST PAS RELIEE
```

Il y a deux façons de mettre des choses ou de placer des valeurs dans les contenants. La première, déjà décrite, est d'utiliser des procédures avec données. La seconde est d'employer la commande RELIE.

```
RELIE "JEAN 25
ECRIS :JEAN
25
```

RELIE est une procédure qui nécessite deux données. Premièrement un mot. Deuxièmement une valeur qui peut être également un mot, ou un nombre, ou une liste. Dans ce cas-ci, RELIE crée un contenant nommé JEAN et y place 25 comme valeur. Remarquez que RELIE n'affiche rien à l'écran. C'est ECRIS qui montre la valeur de JEAN.

Cet exemple illustre bien la différence entre le guillemet et les deux points. La première donnée de RELIE est "JEAN parce que le mot JEAN lui-même est la donnée qui cite à la primitive RELIE le nom qu'elle doit attribuer à la variable. Par ailleurs, la donnée de ECRIS est :JEAN parce qu'on veut que la valeur de JEAN soit affichée.

Voici un autre exemple:

```
RELIE "X "JEAN
ECRIS :X
JEAN
ECRIS :JEAN
25
```

Dans ce cas-ci, RELIE a deux mots cités comme données. La première donnée indique à la primitive RELIE qu'elle doit placer le mot littéral JEAN dans le contenant nommé X. La deuxième donnée, JEAN, est le nom d'un contenant qui avait déjà reçu la valeur 25 par l'emploi préalable de la commande RELIE. Le contenu de JEAN reste donc inchangé.

Différence entre variables globales et variables locales

Lorsque vous créez une variable à l'aide de RELIE alors que Logo est au niveau supérieur, cette variable demeure dans l'espace de travail jusqu'à ce que vous l'effaciez. Il s'agit d'une variable globale. Il existe aussi des variables qui ne restent dans l'espace de travail que le temps de l'exécution d'une procédure. Ce sont des variables locales. Les variables définies comme données de procédures sont toujours des variables locales.

Pour illustrer la différence entre ces deux types de variables, une modification est apportée à ACCUEILLIR de sorte que cette procédure affiche une date.

```
POUR ACCUEILLIR :NOM
EC :DATE
EC "BONJOUR
EC :NOM
EC [PASSEZ UNE BONNE JOURNEE]
FIN
```

Ici, DATE peut être une variable globale non encore définie. Si la procédure ACCUEILLIR est essayée, le message d'erreur suivant apparaîtra:

```
DATE N'EST PAS RELIEE DANS ACCUEILLIR
```

La commande RELIE peut être utilisée au niveau supérieur pour donner une valeur à DATE.

```
RELIE "DATE [14 SEPTEMBRE 1983]
ACCUEILLIR "FRANCOISE
14 SEPTEMBRE 1983
```

```
BONJOUR
FRANCOISE
PASSEZ UNE BONNE JOURNEE
```

Etant une donnée de la procédure, NOM est une variable locale qui ne contient que le mot FRANCOISE pendant que ACCUEILLIR s'exécute.

Il est facile d'oublier que vous avez créé des variables globales. Vous pouvez vérifier si votre espace de travail en contient en utilisant IMNS. Pour les effacer, employez EFN. Voici un exemple démontrant que NOM est bien une variable locale alors que DATE est véritablement globale.

```
RELIE "DATE [2 JUIN 1983]
ACCUELLIR "SYLVIE
2 JUIN 1983
BONJOUR
SYLVIE
PASSEZ UNE BONNE JOURNEE

IMNS
DONNE "DATE [2 JUIN 1983]
```

Aucune valeur n'est affichée pour NOM parce que la variable NOM a disparu une fois terminée l'exécution de la procédure ACCUEILLIR.

Lorsque la commande RELIE est utilisée dans la définition d'une procédure, la variable peut être locale ou globale. Elle est locale s'il s'agit de la donnée d'une procédure qui s'exécute. Si la variable n'est pas une donnée, elle est globale.

Remarquez qu'une procédure n'arrête pas son exécution lorsqu'une sous-procédure est appelée. Ainsi une variable locale à une procédure peut être utilisée par les sous-procédures.

Compréhension d'une ligne Logo

La définition d'une procédure consiste en une suite de lignes d'instructions. Elles sont appelées lignes Logo parce qu'elles peuvent dépasser la longueur d'une ligne d'écran. Par exemple:

```
RELIE "PLUSIEURSNOMS [NORMAND PAUL->
ODETTE RICHARD CHRISTIANNE]
```

La flèche (->) indique que la ligne d'écran qui vient après est la suite de la ligne Logo. Une ligne Logo ininterrompue est obtenue en tapant les mots à la suite les uns des autres sans presser la touche RETOUR. Lorsqu'une ligne Logo dépasse la longueur d'une ligne d'écran, la flèche pointant vers la droite est automatiquement affichée et l'instruction continue de s'inscrire sur la ligne suivante. La ligne Logo se termine dès que la touche RETOUR est pressée.

Voici quelques directives facilitant la compréhension d'une ligne Logo complexe.

1. Lorsque vous voyez le nom d'une procédure, assurez-vous

   a) que vous connaissez son nombre de données;

   b) que vous savez si c'est une commande ou une opération.

2. Le premier mot d'une ligne Logo doit toujours être une commande.

3. Une opération est toujours la donnée d'une autre procédure.

4. Assurez-vous que chaque donnée soit considérée.

5. Quand toutes les données d'une commande ont été considérées, la procédure suivante doit être encore une commande.

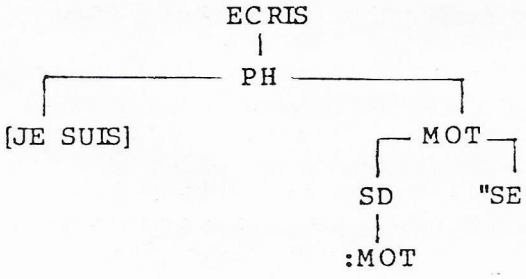
Voici l'exemple d'une ligne Logo complexe. Elle fait partie de la procédure FEMININ qui démontre l'usage de l'opération SD (sauf dernier) au chapitre 2.

ECRIS PH [JE SUIS] MOT SD :MOT "SE

Voyons comment les directives citées plus haut facilitent la compréhension de cette ligne.

ECRIS est une commande qui n'a qu'une donnée. Cette donnée est le résultat de l'opération PH (phrase) qui reçoit deux données. La première est la liste [JE SUIS]. La seconde est ce que donne l'opération MOT qui, à son tour, reçoit deux données. La première doit être l'opération SD qui n'a qu'une donnée :MOT. La deuxième donnée de MOT doit alors être "SE. L'interprétation de la ligne Logo est terminée puisqu'il n'y a plus de noms de procédures et que chaque donnée a été considérée.

Le diagramme suivant illustre cette interprétation.



Par exemple, si le contenu (ou la valeur) de :MOT est AMOUREUX, cette ligne Logo doit provoquer l'affichage de JE SUIS AMOUREUSE.

Description des primitives

Chaque primitive du Logo d'ATARI est décrite dans le reste du présent manuel.

Le nom de la primitive et son abréviation, si elle existe, sont écrites en caractère gras au début de chaque description. Sur la même ligne, on indique si la primitive est une commande, une opération ou une opération infixe. La différence entre une commande et une opération est décrite

sous la rubrique grammaire Logo. L'opération infixe s'insère entres ses données. Les commandes et les opérations ordinaires s'inscrivent avant les données.

Nous poursuivons la description de la primitive en la répétant sur la ligne suivante et en la faisant suivre du type de chaque donnée susceptible de l'accompagner. Toutes les primitives doivent être écrites en lettres majuscules. Toutes les données, dont le type est indiqué en caractères italiques, sont fournies par l'utilisateur.

Suivent des renseignements généraux sur la primitive accompagnés d'exemples d'utilisation.

Description des différentes formes d'une primitive

Si une primitive se présente sous plus d'une forme, chaque forme est indiquée, l'une en dessous de l'autre, en commençant par la plus fréquemment employée. Dans le cas de certaines primitives comme SOMME, le choix arbitraire d'une autre forme est suggéré entre parenthèses. Ceci signifie que cette primitive peut recevoir un nombre illimité d'entrées. Lorsque plus de deux entrées sont utilisées avec cette forme de primitive, vous devez placer une parenthèse gauche avant le nom de la primitive et une parenthèse droite après la dernière donnée.

Il ne faut pas confondre la description du genre de donnée que nécessite une primitive avec la façon dont la donnée elle-même doit être écrite lors de la définition d'une procédure. Les règles qui régissent la définition de procédures ont été décrites sous la rubrique grammaire Logo. Logo interprète une donnée en l'évaluant et la transformant en autre chose. Le tableau 1 montre quels sont ces changements. Par exemple, si on écrit:

```
RELIE :X 22 + 23
```

et que X contient le mot JEAN, alors les vraies données de RELIE sont le mot JEAN et le nombre 45.

Tableau 1

Donnée écrite	Donnée réelle
Mot précédé de guillemets	Mot
Mot précédé de deux points	Contenus du mot qui peuvent être un mot, une liste ou un nombre.
Nombre	Nombre
Liste	Liste
Procédure et ses données	Résultat de procédure qui peut être un mot, une liste ou un nombre.

Dans ce chapitre et dans tout le reste du manuel, on réfère à la donnée réelle lorsqu'on décrit le genre de donnée requis par une primitive. Plusieurs primitives peuvent employer n'importe quel type de donnée, c'est-à-dire que la donnée réelle peut être un mot, une liste ou un nombre. Ceci s'appelle un objet Logo. Si vous cherchez la description de RELIE, vous verrez que sa forme est:

RELIE nom objet

Nom et objet sont deux mots de données. Nom signifie que la première donnée doit être un mot. Un mot est appelé nom s'il devient le nom d'une procédure ou d'une variable. Objet signifie objet Logo. Dans l'exemple donné avant le tableau 1, JEAN est un mot et 45 un objet Logo. Les données sont donc adéquates.

Tous les mots utilisés pour décrire les données des primitives Logo sont expliqués dans les pages suivantes.

Mots de données

caractère	Lettres de l'alphabet, chiffres et signes de ponctuation.
degrés	Degrés d'un angle. Nombre réel entre -9999.9999 et 9999.9999. La commande REPETE peut être utilisée pour dépasser cette limite.
distance	Nombre de -9999.9999 à 9999.9999. La commande REPETE peut être utilisée pour dépasser cette limite.
données	Mots précédés de deux points utilisés conjointement avec POUR.
durée	Entier de Ø à 255.
formespéc	Liste de 16 numéros représentant la grille des formes.
fréq	Entier de 14 à 64,ØØØ en H_Z.
liste	Information contenue entre [ ] crochets.
listeinstruction	Liste de procédures que Logo peut exécuter.
listenom	Liste de noms.
mot	Séquence de caractères sans espaces.
n, a, b, x, y	Nombre.
nom	Mot qui est le nom d'une procédure ou d'une variable.
nomfichier	Nom de fichier (voir le chapitre 1Ø.

numérocond	Entier de Ø à 21 (voir COND et QUAND au chapitre 6).
numérocouleur	Entier de Ø à 127.
numérocrayon	Entier de Ø à 2.
numéroforme	Entier de Ø à 15.
numérolevier	Entier de Ø à 3.
numéromanette	Entier de Ø à 7.
numérotortue	Entier de Ø à 3.
objet	Objet Logo: mot, liste ou nombre.
octet	Unité référentielle de donnée utilisée par l'ordinateur: entier de Ø à 255.
périphérique	Nom de périphérique: "C: pour cassette, "D: pour disque et "P: pour papier. Le " (guillemet) et les : (deux points) sont essentiels.
position, pos	Liste de deux nombres indiquant les coordonnées de la Tortue ou du curseur.
pred	Prédicat, opération dont le résultat est soit le mot VRAI, soit le mot FAUX.
voix	Entier, soit Ø, soit 1.
volume	Entier de Ø à 15.

Chapitre 1

Le graphique Tortue

Lorsque des procédures ou des primitives qui réfèrent à la Tortue sont employées, Logo montre l'écran graphique.

On trouvera ci-après la liste complète des commandes qui changent ce qu'affiche l'écran graphique, ainsi que quelques opérations qui renseignent sur l'état de la Tortue. La plupart de ces primitives sont expliquées dans le manuel Introduction à la programmation via le graphique Tortue.

Le Logo d'ATARI dispose de quatre Tortues capables d'actions dynamiques. Une description sommaire en est donnée dans le manuel ci-haut mentionné.

Les quatre Tortues
Le Logo d'ATARI peut utiliser quatre Tortues à la fois ou chacune d'elles séparément. Quatre primitives permettent de s'adresser à chaque Tortue en particulier: DESIGNE, DEMANDE, CH (chaque), QUI.

Le mouvement dynamique
Il est possible de mettre les Tortues en mouvement à la vitesse désirée par la commande FVIT (fixe vitesse). VIT (vitesse) indique la vitesse à laquelle la ou les Tortues bougent.

La transformation de la Tortue
Les Tortues peuvent changer de couleur et de forme. En plus de leurs formes prédéfinies, les Tortues peuvent subir des transformations illimitées. L'éditeur de forme est utilisé pour dessiner une forme voulue en remplacement de l'originale. La commande EDFOR (édite forme) donne accès à l'éditeur de forme. Les commandes FFOR (fixe forme) et FCT (fixe couleur tortue) déterminent la forme et la couleur de la Tortue utilisée, alors que les opérations FORME et COULEUR renseignent sur ces deux sujets.

La détection d'une collision
Un atout intéressant du Logo d'ATARI est la possibilité de détection d'une collision, ce qui augmente les capacités du graphique Tortue. Les primitives s'y rapportant sont expliquées au chapitre 6: QUAND, CONDP (condition prédicat), COLTC (collision tortue crayon), COLTT (collision tortue tortue); et au chapitre 9: IMD (imprime diablotin), IMDS (imprime diablotins).

AVANCE, AV commande

AVANCE distance

Cette commande fait avancer la Tortue du nombre de pas indiqués par distance dans la direction de son cap. A noter que

l'emploi de AVANCE Ø, alors que le crayon est baissé (BC), affiche un simple point à l'écran à l'endroit où la Tortue se trouve sans faire bouger cette dernière. Il y a erreur si la distance est plus grande que 9999.9999 ou plus petite que -9999.9999.

BC commande

BC

BC (baisse crayon) descend le crayon de la Tortue qui trace alors des lignes de la couleur du crayon utilisé. Au démarrage de Logo, le crayon est baissé.

CAP opération

CAP

Cette opération donne le cap de la Tortue, un nombre plus grand ou égal à Ø et plus petit que 36Ø. Logo suit le système de la boussole où le cap Ø indique le nord, 9Ø l'est, 18Ø le sud et 27Ø l'ouest. Au démarrage de Logo, le cap de la Tortue est à Ø pointant vers le haut.

nord
Ø°

ouest 27Ø° 9Ø° est

18Ø°
sud

CC opération

CC numérocrayon

CC (couleur crayon) donne un nombre représentant la couleur du numéro du crayon utilisé, soit Ø, 1 ou 2. Au démarrage de Logo, CC Ø est 15 (or), CC 1 est 47 (pourpre) et CC 2 est 121 (orange).

CH commande

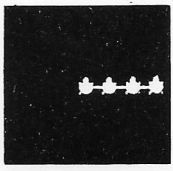
CH listeinstruction

CH (chaque) indique aux Tortues utilisées d'effectuer, chacune

séparément, les commandes comprises dans la listeinstruction. S'il y a plus d'une Tortue en activité, la première Tortue exécute toutes les commandes de la listeinstruction avant que la deuxième Tortue ne fasse quoi que ce soit. Cette commande est particulièrement utile lorsque vous voulez que chaque Tortue accomplisse quelque chose de légèrement différent des autres.

Exemples:

Les instructions suivantes alignent les Tortues à 20 pas de distance les unes des autres et fixent la couleur de chacune d'elles selon son propre numéro:



```
DESIGNE [0 1 2 3]
ORIGINE
CH [FX QUI * 20]
CH [FCT QUI * 8]
```

QUI donne le numéro identifiant chaque Tortue. Ainsi, la Tortue 0 effectue FX (fixe X) 0 et FCT (fixe couleur tortue) 0; la Tortue 1 effectue FX 20 et FCT 8 et ainsi de suite.

CH, tout comme DEMANDE, n'influence pas le choix de la ou des Tortues qui est (sont) en train d'exécuter une directive. DEMANDE fait agir toutes les Tortues en même temps, alors que CH ne fait agir qu'une Tortue à la fois. Les exemples suivants illustrent ceci:

```
POUR PREPARE
VE DESIGNE [0 1 2 3]
CH [DR 90 * QUI]
FIN
```

```
PREPARE
DEMANDE [Ø 1 2 3] [REPETE 4 [AV 5Ø D->
R 9Ø]]
PREPARE
CH [REPETE 4 [AV 5Ø DR 9Ø]]
```

COULEUR opération

COULEUR

Cette opération donne un nombre entier de Ø à 127 représentant la couleur actuelle de la Tortue. Lorsque Logo est mis en marche, la Tortue Ø est blanche (7), la Tortue 1 est orangée (2Ø), la Tortue 2 est pourpre (44) et la Tortue 3 est bleue (68). Voir FOND pour le tableau des couleurs et FCT (fixe couleur tortue) pour le changement de la couleur des Tortues.

CRAYON opération

CRAYON

Le résultat de cette opération est un mot qui décrit l'état actuel du crayon de la Tortue: BC, LC, GC, IC. Pour plus de détails, voir la description de chacune de ces primitives. Lorsque Logo est mis en marche, le résultat de CRAYON est BC.

CT commande

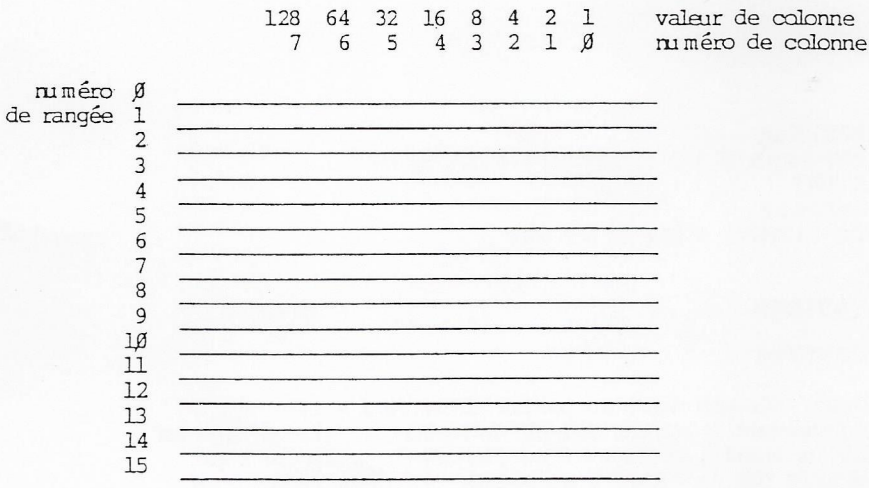
CT

CT (cache tortue) rend la Tortue invisible mais quand même capable de dessiner.

DECRISFOR opération

DECRISFOR numéroforme

DECRISFOR (décris forme) donne une liste de 16 numéros représentant la grille de numéroforme, chacun étant un nombre entier de 1 à 15. Numéroforme ne peut pas être Ø. Chaque forme est constituée par une grille de 8 colonnes et 16 rangées. Chaque élément de la liste est la somme de la valeur des bits d'une rangée.



Chaque colonne a un numéro et une valeur. La valeur de chaque colonne, numérotée de Ø à 7, progresse par puissance de 2. Par exemple, la colonne numéro 3 ayant la valeur 8, la colonne numéro 4 aura la valeur 16, c'est-à-dire 2 à la quatrième puissance.

Le premier élément de la liste correspond à la première rangée de la forme. Si toute la rangée est remplie, ce numéro est 255, la somme de toutes les valeurs des colonnes. Chaque somme possible est unique. Si la dernière case seulement de la rangée est remplie, la valeur de la somme est 1. Si seulement les cinquième et sixième cases à partir de droite sont remplies, la valeur est 48, soit la somme des valeurs des colonnes 4 et 5.

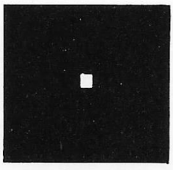
DECRISFOR est surtout utile pour mettre les formes créées en réserve sur disquette ou sur cassette. Il faut d'abord transposer les formes en variables puis sauver l'espace de travail. Pour plus de détails, voir le chapitre 16 de l'Introduction à la programmation via le graphique Tortue.

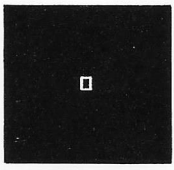
Exemples:

Supposons que la forme numéro 1 est une boîte pleine et que la forme numéro 2 en est le contour:

```
EC DECRISFOR 1
255 255 255 255 255 255 255 255 2->
55 255 255 255 255 255 255
```

```
EC DECRISFOR 2
255 129 129 129 129 129 129 129 1->
29 129 129 129 129 129 255
```





DEFFOR commande

DEFFOR numéroforme numérospéc

DEFFOR (définis forme) associe au numéroforme la forme définie par numérospéc qui en indique les spécifications. Le résultat de l'opération DECRISFOR peut être l'entrée numérospéc dans DEFFOR. La commande DEFFOR permet de définir des formes à l'intérieur d'un programme plutôt que de se servir de l'éditeur de forme.

Exemples:

La procédure REMPLACE peut modifier une rangée dans une forme déjà définie.

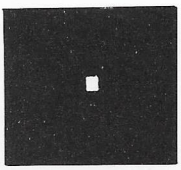
```
POUR REMPLACE :POS :NOUVR :FORME
SI :POS = 1 [RT PH :NOUVR SP :FORME]
RT PH PREMIER :FORME REMPLACE :POS - ->
1 :NOUVR SP :FORME
FIN

EC DECRISFOR 1
255 255 255 255 255 255 255 255 255 2->
55 255 255 255 255 255 255
```

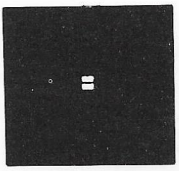
Numéroforme 1 est une boîte pleine.

```
DEFFOR 1 REMPLACE 8 Ø DECRISFOR 1
```

coupe la boîte en son centre.



BOITE PLEINE



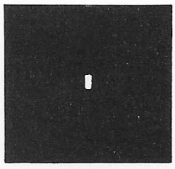
BOITE COUPEE

```
POUR CHANGEFOR :FORME :POS :N
SI :POS > 16 [RT :FORME]
RT PH (PREMIER :FORME) - :N CHANGEFOR->
 SP :FORME :POS + 1 :N
FIN
```

Si numéroforme 1 est une boîte pleine,

```
DEFFOR 1 CHANGEFOR DECRISFOR 1 1 15
```

diminue de moitié la dimension de chaque rangée.



DEMANDE commande ou opération

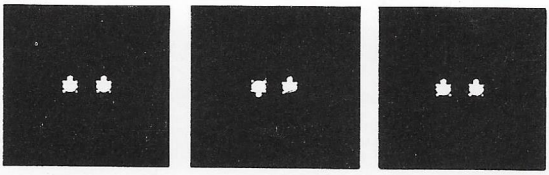
DEMANDE numérotortue listeinstruction
DEMANDE listenumérotortue listeinstruction

Cette primitive demande à la ou aux Tortues, spécifiées à la première entrée par le numérotortue ou la listenumérotortue, d'exécuter les instructions de la deuxième entrée. Ceci n'influence pas les Tortues qui sont en train d'exécuter des directives. Ces Tortues obéissent déjà à la commande DESIGNE. Si la listeinstruction est une opération, DEMANDE donne le résultat de l'opération. Le numérotortue est un entier de Ø à 3.

Exemple:

Les instructions suivantes disent à la Tortue 2 de pointer dans la même direction que la Tortue 1.

```
DESIGNE 1 MT
DESIGNE 2 MT
EC QUI
2
LC FPOS [-3Ø Ø]
FCAP 18Ø
FCAP DEMANDE 1 [CAP]
EC QUI
2
```



DESIGNE commande

DESIGNE numérotortue
DESIGNE listenumérotortue

Cette commande désigne la ou les Tortues qu'on désire utiliser. Si la commande DESIGNE n'est pas employée, les instructions sont transmises à la Tortue numéro Ø.

La première fois qu'on s'adresse aux autres Tortues à l'aide de DESIGNE après le démarrage de Logo, elles apparaissent sur l'écran sans qu'il soit nécessaire de commander MT (montre tortue). C'est la seule fois où DESIGNE a cet effet.

Exemples:

Les instructions suivantes donnent la couleur rouge (4Ø) à la Tortue numéro 3, et la couleur bleue (7Ø) aux Tortues 1, 2 et Ø.

```
DESIGNE 3
FCT 4Ø
DESIGNE [1 2 Ø]
FCT 7Ø
DESIGNE [0 1 2 3]
CH [AV 30 * QUI]
```

DESIGNE peut aussi accepter une liste du même numéro de Tortue:

```
DESIGNE [Ø Ø Ø Ø]
AV 1Ø
```

Dans ce cas, AV 1Ø est répété quatre fois.

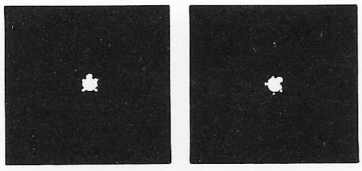
DROITE, DR commande

DROITE degrés

Cette commande fait tourner la Tortue à droite, dans le sens des aiguilles d'une montre, du nombre de degrés indiqués. Il y a erreur si les degrés sont plus grands que 9999.9999 ou plus petits que -9999.9999.

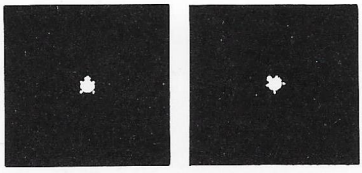
Exemples:

DROITE 45 (fait tourner la Tortue de 45 degrés vers la droite)



DR 45

DROITE -45 (fait tourner la Tortue de 45 degrés vers la gauche)



DR -45

EDFOR commande

EDFOR numéroforme

EDFOR (édite forme) démarre l'éditeur Logo qui permet la transformation des Tortues. EDFOR ramène dans l'éditeur la forme correspondant au numéroforme lequel est un nombre entier de 1 à 15. A noter que Ø étant le numéro de la Tortue originale, cette forme ne peut pas être éditée. Voir la fin de

ce chapitre pour plus d'informations sur l'éditeur de forme de la Tortue.

ENROULE commande

ENROULE

La Tortue qui obéit à cette commande s'enroule autour de l'écran, c'est-à-dire que si elle diparaît par un côté de l'écran, elle réapparaît par le côté opposé, ne quittant ainsi jamais les limites visibles de l'écran. Le champ de la Tortue devient alors illimité et elle peut s'y mouvoir en exécutant AV (avance) ou RE (recule) d'un nombre infini de pas.

Lorsqu'on donne la commande ENROULE, l'écran se vide. Voir également FENETRE.

Exemple:

```
ENROULE
DR 5
AV 5ØØ
EC POS
43.57787 -77.3ø265
```

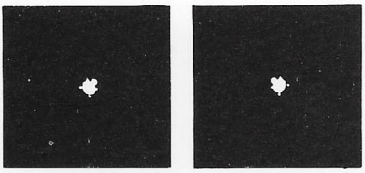
FCAP commande

FCAP degrés

FCAP (fixe cap) fait pivoter la Tortue de telle sorte que son cap se fixe aux degrés indiqués. Les nombres positifs la font tourner dans le sens des aiguilles d'une montre. DROITE et GAUCHE orientent la Tortue vers un cap relatif à son cap actuel, tandis que FCAP l'oriente vers un cap absolu, indépendamment de son cap actuel. Il y a erreur si les degrés sont plus grands que 9999.9999 ou plus petits que -9999.9999. Voir CAP.

Exemples:

```
FCAP 45        Oriente la Tortue en direction nord-est.
FCAP -45       Oriente la Tortue en direction nord-ouest.
EC CAP
315
```



FCAP 45 FCAP -45

FCC commande

FCC numérocrayon numérocouleur

FCC (fixe couleur crayon) fixe le numérocrayon (Ø, 1 ou 2) au numérocouleur indiqué qui est un entier de Ø à 127. Il est possible de changer la couleur d'une forme déjà dessinée en changeant le numéro du crayon ou en assignant un nouveau numérocouleur au numérocrayon déjà employé.

Avant d'utiliser FCC, on doit assigner un numérocrayon à FNC (fixe numéro crayon) à moins qu'on ne change le numéro du crayon utilisé.

Exemple:

```
REPETE 4 [AV 2Ø DR 9Ø]
FCC Ø 12Ø
```

Si l'exemple ci-haut est expérimenté au démarrage de Logo, le carré passe du ton or au ton orange.

FCT commande

FCT numérocouleur

FCT (fixe couleur tortue) fixe la couleur de la Tortue au numérocouleur indiqué qui doit être un entier de Ø à 127.

FENETRE commande

FENETRE

Cette commande rend le champ de la Tortue très vaste. Une portion seulement de ce champ est visible à l'écran comme si on regardait par une petite fenêtre. En réalité, lorsque la Tortue disparaît, elle n'est pas cachée mais simplement en dehors des limites de l'écran.

Le champ entier de la Tortue a 25119 pas en hauteur et 19841 pas en largeur. Pour atteindre les limites de FENETRE il faut répéter AV (avance) ou RE (recule) un certain nombre de fois avec son entrée la plus grande possible soit 9999.9999. Lorsque la commande FENETRE est utilisée, l'écran se vide. Voir aussi ENROULE.

Pour restreindre le champ de la Tortue, on doit se servir de la détection de collision.

Exemple:

```
FENETRE
DR 5
AV 5ØØ
EC POS
43.57787 498.Ø9735
```

FFOND commande

FFOND numérocouleur

FFOND (fixe fond) fixe la couleur du fond au numérocouleur indiqué. Il y a 128 couleurs de fond numérotées de Ø à 127.

Exemple:

La procédure suivante montre les 128 couleurs de fond possibles:

```
POUR CHANGEFOND
SI FOND = 127 [FFOND Ø ATTENDS 3Ø]
FFOND 1 + FOND
EC FOND ATTENDS 3Ø
CHANGEFOND
FIN
```

CHANGEFOND

Pour arrêter cette procédure on doit presser la touche ARRET.

FFOR commande

FFOR numéroforme

FFOR (fixe forme) détermine la forme de la Tortue utilisée en lui donnant le numéroforme indiqué qui doit être un entier de Ø à 15. Les formes sont créées en employant les primitives EDFOR ou DEFFOR. On ne peut pas modifier la forme Ø qui est celle de la Tortue. Les formes numérotées de 1 à 15 sont vides au démarrage de Logo. Pour plus de détails à ce sujet, voir l'éditeur de forme de Tortue à la fin du présent chapitre.

Exemple:

Si les Tortues ont été transformées, les commandes suivantes leur redonnent leur forme originale:

```
DESIGNE [Ø 1 2 3] FFOR Ø
```

FNC commande

FNC numérocrayon

FNC (fixe numéro crayon) fixe au numérocrayon indiqué le crayon utilisé par la ou les Tortues immédiatement concernées. Trois numéros de crayons peuvent être employés: Ø, 1, 2. Le crayon avec lequel la Tortue dessine est ainsi déterminé. On utilise FCC (fixe couleur crayon) pour déterminer la couleur du crayon. Quand Logo démarre, la ou les Tortues utilisent le crayon numéro Ø.

FOND opération

FOND

Le résultat de cette opération est un nombre représentant la couleur actuelle du fond. Lorsque Logo démarre, FOND est bleu clair (74). Pour de plus amples détails sur la façon de régler la couleur du fond, voir FFOND (fixe fond).

L'ordinateur ATARI vous propose 128 couleurs. Elles sont regroupées en 16 couleurs principales ayant chacune 8 tons codés comme suit:

```
 Ø -   7 gris
 8 -  15 orange clair (or)
16 -  23 orange
24 -  31 rouge orangé
32 -  39 rose
40 -  47 pourpre
48 -  55 pourpre bleuté
56 -  63 bleu
64 -  71 bleu
72 -  79 bleu clair
8Ø -  87 turquoise
88 -  95 vert bleuté
96 - 1Ø3 vert
1Ø4 - 111 jaune verdâtre
112 - 119 orange verdâtre
12Ø - 127 orange clair
```

Pour chaque couleur, le nombre le moins élevé correspond au ton le plus foncé, et le nombre le plus élevé au ton le plus

clair. Par exemple, Ø est noir et 7 est blanc.

Note: Les couleurs peuvent varier selon le type de téléviseur ou de moniteur, ou encore selon l'état de fonctionnement de l'appareil ou des possibilités de réglage de la couleur. Les couleurs des sytèmes PAL peuvent différer de celles énumérées ci-haut.

FORME opération

FORME

Le résultat de cette opération donne un nombre représentant la forme de la Tortue utilisée. La forme originale de la Tortue porte le numéro Ø. Les numéros des formes diffèrent des numéros des Tortues.

Exemple:

```
DEFFOR 12 [255 255 255 255 255 255 2->
55 255 255 255 255 255 255 255 25->
5]
DESIGNE 3 FFOR 12
EC FORME
12
EC QUI
3
```

FPOS commande

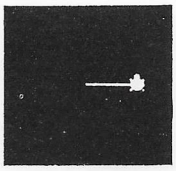
FPOS position

FPOS (fixe position) déplace la Tortue à la position indiquée par une liste de deux nombres, les coordonnées X et Y. Voir POS (position) pour plus de détails. X et Y prennent comme entrée maximale 9999.9999, que la Tortue évolue dans le champ FENETRE ou dans le champ ENROULE. Si le crayon de la Tortue est baissé, elle laisse une trace à partir de sa position antérieure jusqu'à sa nouvelle position.

Exemple:

FPOS [8Ø Ø]

déplace la Tortue à mi-chemin entre le centre et le bord droit de l'écran.



FPOS [8Ø Ø]

FVIT commande

FVIT vitesse

FVIT (fixe vitesse) règle la vitesse de la Tortue sans changer son cap. Si la vitesse est plus grande que Ø, la Tortue avance; si elle est inférieure à Ø, la Tortue recule. La Tortue ne bouge pas si la vitesse est égale à Ø. Il y a erreur si la vitesse est plus grande que 199 ou inférieure à -199. L'entrée de FVIT peut être un nombre réel.

Exemple:

La procédure suivante déplace chaque Tortue en direction est à une vitesse au hasard entre 1 et 3Ø:

```
POUR EST
DESIGNE [Ø 1 2 3] MT
FCAP 9Ø
CH [FVIT 1 + HASARD 3Ø]
FIN
```

FX commande

FX x

FX (fixe X) déplace la Tortue horizontalement jusqu'au point d'abscisse X (l'ordonnée Y reste inchangée). Si son crayon est baissé, la Tortue trace une ligne horizontale.

FX -158

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

FX -158 FX 2 * XCOR

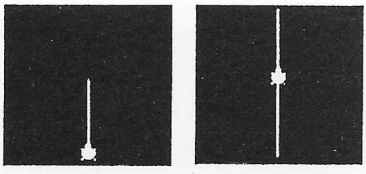
FY commande

FY Y

FY (fixe Y) déplace la Tortue verticalement jusqu'au point d'ordonnée Y (l'abscisse X ne change pas).

FY - 95

déplace la Tortue verticalement jusqu'au bord inférieur de l'écran.



FY -95 FY 2 * YCOR

GAUCHE, GA commande

GAUCHE degrés

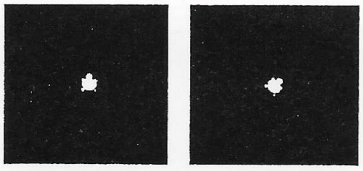
Cette commande fait tourner la Tortue en sens inverse des aiguilles d'une montre du nombre de degrés indiqué. Il y a erreur si le nombre de degrés est plus grand que 9999.9999 ou plus petit que -9999.9999.

Exemples:

GAUCHE 45 (fait tourner la Tortue de 45 degrés vers la gauche)

GAUCHE 45

GAUCHE -45 (fait tourner la Tortue de 45 degrés vers la droite)



GAUCHE -45

GC commande

GC

GC (gomme crayon) transforme le crayon de la Tortue en gomme à effacer. La Tortue efface alors toute ligne sur laquelle elle passe. Les primitives suivantes changent cet état du crayon: BC, LC, IC.

IC commande

IC

IC (inverse crayon) a pour effet, lorsque la Tortue se déplace, d'effacer les lignes déjà tracées et de tracer des lignes là où il n'y en a pas. Les effets de cette inversion sont complexes. Ils dépendent de la couleur du fond, de celle du crayon, de l'horizontalité ou de la verticalité des lignes. Les meilleurs résultats sont obtenus sur fond noir. Les primitives BC, LC ou GC annulent l'effet de IC.

IC agit avec FVIT mais les résultats obtenus sont très inconsistants. Il n'est donc pas recommandé d'utiliser ces deux primitives ensemble.

LC commande

LC

LC (lève crayon) remonte le crayon de la Tortue qui ne laisse ainsi plus de trace en se déplaçant.

MT commande

MT

MT (montre tortue) rend la Tortue visible. Voir aussi CT (cache tortue). Si la forme de la Tortue n'est pas définie, c'est-à-dire que si DECRISFOR retourne une liste de zéros, MT ne rend pas la Tortue visible.

NC opération

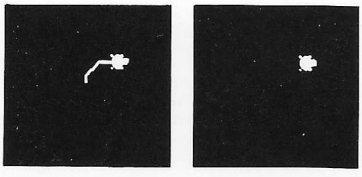
NC

NC (numéro crayon) donne un entier (Ø, 1 ou 2) représentant le numéro du crayon utilisé. Les Tortues d'ATARI peuvent employer l'un des trois crayons pour dessiner. A la mise en marche de Logo, NC est Ø. La commande FNC (fixe numéro crayon) indique à la Tortue quel crayon choisir. La couleur de chaque crayon peut être modifiée par la commande FCC.

NETTOIE commande

NETTOIE

Cette primitive enlève tout tracé sur l'écran sans changer l'état de la Tortue qui, elle, reste visible tout comme le texte.

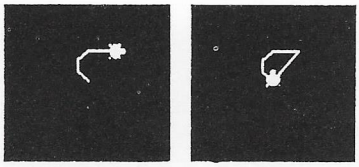


NETTOIE

ORIGINE commande

ORIGINE

L'effet de la commande ORIGINE est de ramener la Tortue au centre de l'écran en fixant son cap à Ø. Cette commande équivaut à FPOS [Ø Ø] FCAP Ø. Si le crayon de la Tortue est baissé, elle trace une ligne en se déplaçant de l'endroit où elle se trouve jusqu'à son point d'origine.



ORIGINE

POS opération

POS

POS (position) donne la liste des coordonnées [X Y] de la position actuelle de la Tortue. Quand Logo démarre, la Tortue est à [Ø Ø], au centre de l'écran. Voir FPOS (fixe position) pour d'autres détails sur la position de la Tortue.

120

-158 Ø 161

Le rapport d'aspect peut varier d'un appareil à l'autre. Le graphique ci-contre correspond au ratio .FRATIO .8.

-119

QUI opération

QUI

Cette opération retourne le ou les numéros des Tortues utilisées. Il s'agit d'un entier de Ø à 3 représentant les quatre Tortues accessibles au Logo d'ATARI.

QUI est utile avec la commande CH (chaque) lorsqu'on veut que les Tortues effectuent en même temps des instructions différentes.

Exemples:

```
DESIGNE [1 2]
EC QUI
1 2
```

Les instructions suivantes font avancer simultanément les quatre Tortues, chacune dans une direction différente:

```
DESIGNE [Ø 1 2 3]
ORIGINE
CH [FCAP 9Ø * QUI]
AV 3Ø
```

RECULE, RE commande

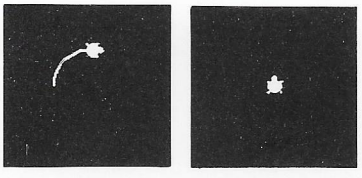
RECULE distance

Cette primitive sert à déplacer la Tortue vers l'arrière, du nombre de pas indiqué par distance. Son cap ne change pas. A noter que si l'on commande RECULE Ø alors que le crayon est baissé (BC), un point apparaît à l'endroit où se trouve la Tortue, sans que cette dernière se déplace. Il y a erreur si distance est plus grand que 9999.9999 ou plus petit que -9999.9999.

VE commande

VE

La commande VE (vide écran) efface tout tracé sur l'écran et place la ou les Tortues à la position [Ø Ø] au centre de l'écran, pointant vers le nord, cap Ø. VE annule également toute action du ou des QUAND Diablotins. Voir QUAND au chapitre 6. VE n'efface pas le texte. Voir VT au chapitre 8.



VE

VISIBLEP opération

VISIBLEP

VISIBLEP (visible prédicat) retourne VRAI si la Tortue est visible, FAUX si elle ne l'est pas. Logo considère la Tortue visible même lorsque l'observateur ne peut la voir parce qu'elle est en dehors de l'écran. Par exemple, si le champ de la Tortue devient très vaste par l'emploi de FENETRE, la Tortue peut être hors de vue mais VISIBLEP indique VRAI. En réalité, la Tortue n'est pas cachée et serait visible si ce n'était des limites de l'écran.

VIT opération

VIT

VIT (vitesse) indique la vitesse à laquelle se déplace la Tortue utilisée. VIT est définie en pas de Tortue par 1/60èmes de seconde.

VIT peut ne pas donner la vitesse exacte assignée préalablement par l'entrée de FVIT. Ceci est dû à la façon dont Logo calcule.

Exemple:

La procédure suivante arrête toute Tortue qui enfreint la "limite de vitesse":

```
POUR ENFREINDRE :LIMITEVIT
CH [SI VIT > :LIMITEVIT [FVIT Ø]]
FIN

DESIGNE [Ø 1 2 3]
CH [LC FVIT 1Ø + 2Ø * QUI]
ENFREINDRE 4Ø
```

La primitive CH (chaque) est employée pour que Logo vérifie la vitesse de chaque Tortue.

XCOR opération

XCOR

XCOR (X coordonnée) retourne l'abscisse de la position actuelle de la Tortue.

YCOR opération

YCOR

YCOR (Y coordonnée) retourne l'ordonnée de la position actuelle de la Tortue.

Exemples:

```
VE EC YCOR
Ø
AV 9Ø
EC YCOR
9Ø
```

La procédure suivante donne le sinus d'un angle. Ceci équivaut

au résultat de l'opération qu'effectue la primitive SIN.

```
POUR DONNESIN :ANGLE
ORIGINE
FCAP 9Ø
GA :ANGLE
AV 1ØØ
RT YCOR / 1ØØ
FIN

EC DONNESIN 3Ø
Ø.5ØØØØ2136 4
```

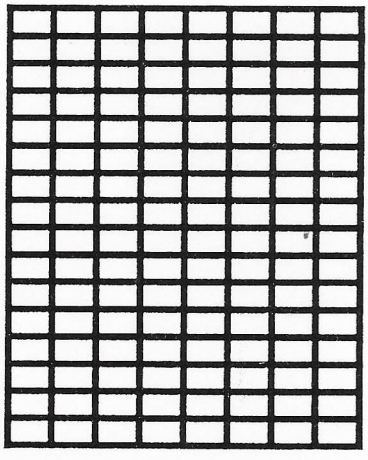
L'éditeur de forme

On peut créer un nombre illimité de formes en se servant de l'éditeur de forme. Cependant, il n'y a que 16 formes de Tortues accessibles à la fois. Seule la forme numéro Ø représentant la forme originale de la Tortue ne peut être modifiée.

EDFOR est la commande qui démarre l'éditeur de forme Logo. Son entrée est un numéroforme entier de 1 à 15. Ces formes se présentent comme une grille vide quand Logo commence. EDFOR amène dans l'éditeur de forme le numéroforme indiqué. Lorsqu'un numéroforme est défini pour la première fois, sa grille présente 128 cases vides constituées de 8 colonnes et 16 rangées.

Exemple:

```
EDFOR 1
```



Il n'y a pas de symbole d'invite mais le curseur apparaît au coin supérieur gauche de l'écran et peut être déplacé dans n'importe quelle case. On se sert de la barre d'espacement pour remplir ou vider les cases désirées pour créer une nouvelle forme.

Comment modifier les formes en déplaçant le curseur

Pour déplacer le curseur sans changer la forme, on doit utiliser la combinaison de la touche CTRL avec les touches fléchées. Pour modifier une case recouverte par le curseur, on presse sur la barre d'espacement, ce qui vide une case pleine et en remplit une vide. Ceci définit la forme. Il faut placer le curseur sur la case à changer avant de presser la barre d'espacement. Les combinaisons de touches suivantes opèrent ces déplacements:

CTRL —> déplace le curseur d'une case vers la droite;

CTRL <— déplace le curseur d'une case vers la gauche;

CTRL ↑ déplace le curseur d'une case vers le haut;

CTRL ↓ déplace le curseur d'une case vers le bas.

Il ne faut pas presser la touche ATARI (▲) ou la touche vidéo inverse (■) lorsque l'éditeur de forme est en marche car ceci rend la barre d'espacement inopérante. Pour annuler cet effet, pressez à nouveau cette touche.

Comment quitter l'éditeur Logo

Pour quitter l'éditeur Logo, on doit presser soit la touche QUITTE, soit la touche ARRET.

QUITTE — quitte l'éditeur de forme en sauvant les changements apportés;

ARRET — arrête l'éditeur de forme sans sauver les changements apportés.

Voir le chapitre 16 du manuel Introduction à la programmation via le graphique Tortue pour un exemple d'édition de forme.

Chapitre 2

Mots et listes

Dans le langage Logo, il y a deux types d'objets: les mots et les listes. Les primitives servent soit à les rassembler, à les démonter, ou à les examiner.

Un mot est composé de caractères.

Exemples:

```
BONJOUR
X
314
3.14
R2D2
COCHOND'INDE
COCHON.D'INDE
HEN3RI
QUI?
!COMMENT!
```

sont tous des mots. Chaque caractère est un élément du mot. Le mot HEN3RI contient 6 éléments.

```
H E N 3 R I
```

Un mot est généralement délimité par des espaces. Cela signifie qu'il y a un espace avant et un espace après, ce qui 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 "\".

Exemple:

```
EC "COCHON\-D'INDE
COCHON-D'INDE
```

Le guillemet (") et les deux points (:) ne sont pas des délimiteurs de mots.

Une liste est composée d'objets Logo. Chacun d'eux peut être un mot ou une autre liste. Une liste s'inscrit entre 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?

La liste [1 [1 2] [17 [17 2]]] ne contient que 3 éléments et non pas six, 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 qui est un mot sans élément. Vous le tapez au clavier en tapant un guillemet, ", suivi d'un espace. Voir VIDEP pour des exemples de liste vide et de mot vide.

Les opérations PREMIER, SP (sauf premier), DERNIER et SD (sauf dernier) sont utilisées pour démonter les mots et les listes. Le tableau suivant en montre le fonctionnement. Pour expérimenter ces exemples d'opérations, il faut employer la commande ECRISC (écris complet) ou son abréviation ECC.

Opération	Entrée	Sortie
PREMIER	"JEAN	J
SP	"JEAN	EAN
PREMIER	[MARIE JEAN ANDRE]	MARIE
SP	[MARIE JEAN ANDRE]	[JEAN ANDRE]
PREMIER	[[MARIE JEAN] ANDRE]	[MARIE JEAN]
SP	[[MARIE JEAN] ANDRE]	[ANDRE]
PREMIER	[ ] ou "	erreur
SP	[ ] ou "	erreur

DERNIER et SD (sauf dernier) fonctionnent de la même manière mais isolent le dernier élément.

Pour rassembler les mots et les listes, Logo utilise cinq opérations: MP (mets premier), MD (mets dernier), LISTE, PH (phrase) et MOT. Le tableau suivant compare ces cinq primitives:

Opération	Entrée 1	Entrée 2	Sortie
MP	"PLAISIR	"LOGO	erreur
LISTE	"PLAISIR	"LOGO	[PLAISIR LOGO]
MD	"PLAISIR	"LOGO	erreur
PH	"PLAISIR	"LOGO	[PLAISIR LOGO]
MOT	"PLAISIR	"LOGO	PLAISIRLOGO

```
MP        "LOGO   [EST ENRICHISSANT]    [LOGO EST ENRICHISSANT]
LISTE     "LOGO   [EST ENRICHISSANT]    [LOGO [EST ENRICHISSANT]]
MD        "LOGO   [EST ENRICHISSANT]    [EST ENRICHISSANT LOGO]
PH        "LOGO   [EST ENRICHISSANT]    [LOGO EST ENRICHISSANT]
MOT       "LOGO   [EST ENRICHISSANT]    erreur

LISTE     [ET ENCORE]  [A VENIR]  [[ET ENCORE] [A VENIR]]
MD        [ET ENCORE]  [A VENIR]  [A VENIR [ET ENCORE]]
PH        [ET ENCORE]  [A VENIR]  [ET ENCORE A VENIR]
MOT       [ET ENCORE]  [A VENIR]  erreur

MP        "ORDINATEUR     []    [ORDINATEUR]
LISTE     "ORDINATEUR     []    [ORDINATEUR []]
MD        "ORDINATEUR     []    [ORDINATEUR]
PH        "ORDINATEUR     []    [ORDINATEUR]
MOT       "ORDINATEUR     []    erreur
```

ASCII opération

ASCII caractère

Retourne le code ASCII de caractère. L'appendice F donne une table de tous les codes ASCII. Si l'entrée contient plus d'un caractère, ASCII n'opère que sur le premier. Voir aussi CAR.

Exemple:

La procédure CODESECRET définit un nouveau mot en utilisant le code chiffré de César qui ajoute 3 à chaque lettre.

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

POUR CODE :LETTRE
RELIE "NUMERO (ASCII :LETTRE) + 3
SI :NUMERO > ASCII "Z [RELIE "NUMERO ->
:NUMERO - 26]
RETOURNE CAR :NUMERO
FIN

EC CODESECRET "CHAT
FKDW

EC CODESECRET "CRAYON
FUDBRQ
```

CAR opération

CAR n

CAR (caractère) donne le caractère dont le code ASCII est n, un entier de Ø à 255. L'appendice F présente le tableau des codes ASCII.

Les codes ASCII sont organisés de la manière suivante:

```
  Ø -  31 caractères graphiques
 32 -  47 ponctuation
 48 -  57 chiffres
 58 -  64 ponctuation
 65 -  9Ø majuscules
 91 -  96 ponctuation
 97 - 122 minuscules
123 - 127 caractères graphiques
128 - 255 vidéo inverse des caractères Ø à 127
```

Exemple:

```
POUR MINUSCULE :LETTRE
RELIE "MIN 32 + ASCII :LETTRE
SI ET :MIN > 96 :MIN < 123 [RT CAR :M->
IN] [RT :LETTRE]
FIN
```

Cette procédure retourne une lettre minuscule. Si on donne à cette procédure un autre caractère qu'une lettre, elle retourne ce même caractère:

```
EC MINUSCULE "A
a

EC MINUSCULE "8
8
```

COMPTE opération

COMPTE objet

Cette opération compte le nombre d'éléments d'un mot ou d'une liste.

Exemples:

```
EC COMPTE [VOYEZ LE BRICK]
3

EC COMPTE [[VOYEZ LE BRICK] GEANT]
2

EC COMPTE "LOGO
4
```

```
RELIE "CLASSE [FRANCOISE MICHEL ALAI->
N 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 ELEM (1 + HASARD COMPTE :LISTE) :L->
ISTE
FIN

POUR ELEM :N :OBJET
SI :N = 1 [RT PREMIER :OBJET]
RT ELEM :N - 1 SP :OBJET
FIN

PIQUEHASARD :CLASSE          Voir la liste CLASSE plus haut.
MARIE-PAULE

PIQUEHASARD "LOGO
G
```

DERNIER opération

DERNIER objet

Transmet le dernier élément d'objet. DERNIER appliqué au mot vide ou à la liste vide est une erreur.

Exemples:

```
ECC DERNIER [MARIE SEBASTIEN ONESIME->
]
ONESIME

ECC DERNIER "CANNELLE
E

ECC DERNIER [[LE] SOLDAT VA [PARTIR ->
SAUTER DORMIR]]
[PARTIR SAUTER DORMIR]
```

La procédure suivante épelle un mot à l'envers:

```
POUR INVEPELLE :ENTR
SI VIDEP :ENTR [STOP]
TAPE DERNIER :ENTR
INVEPELLE SD :ENTR
FIN
```

```
INVEPELLE "CHOCOLAT
TALOCOHC
```

EGALP opération

EGALP objet1 objet2

EGALP (égal prédicat) retourne VRAI si objet1 et objet2 sont soit des nombres égaux, des mots ou des listes identiques; sinon, retourne FAUX. Cette primitive équivaut au symbole =, opération infixe.

Exemples:

```
EC EGALP "ROUGE PREMIER [ROUGE JAUNE->
]
VRAI

EC EGALP 1ØØ 5Ø * 2
VRAI

EC EGALP [LE CHAT GRIS] [LE CHAT]
FAUX

EC EGALP " []                  Le mot vide et la liste vide
FAUX                           ne sont pas identiques.
```

L'opération suivante indique le rang que la première entrée occupe dans la seconde et retourne Ø si la première entrée n'est pas un élément de la seconde:

```
POUR RANG :UN :TOUT
SI VIDEP :TOUT [RT Ø]
SI EGALP :UN DERNIER :TOUT [RT COMPTE->
 :TOUT]
RT RANG :UN SD :TOUT
FIN

EC RANG "DEUX [UN DEUX TROIS]
2

EC RANG "A "AMOUR
1
```

LISTE opération

LISTE objet1 objet2

Cette opération fournit une liste dont les éléments sont objet1 et objet2. Chaque entrée de LISTE peut être un mot ou une liste.

Exemples:

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

ECC LISTE [LA RAISON DU PLUS FORT] [->
EST TOUJOURS LA MEILLEURE]
[[LA RAISON DU PLUS FORT] [EST TOUJOU->
RS LA MEILLEURE]]
```

LISTEP opération

LISTEP objet

LISTEP (liste prédicat) constitue VRAI si objet est une liste; sinon, constitue FAUX.

Exemples:

```
EC LISTEP 3
FAUX

EC LISTEP [3]
VRAI

EC LISTEP []
VRAI

EC LISTEP "
FAUX

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

EC LISTEP SP "CHOCOLAT
FAUX

EC LISTEP SP [CHOCOLAT]
VRAI
```

MD opération

MD objet liste

MD (mets dernier) retourne une nouvelle liste qui ajoute objet comme dernier élément de liste. Voir le tableau du début de ce chapitre qui compare MD aux autres primitives combinant mots et listes.

Exemple:

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

```
POUR NOUVEAUMOT :INFO
```

```
RELIE "DICTIONNAIRE MD :INFO :DICTION->
NAIRE
FIN

RELIE "DICTIONNAIRE [[MAISON CASA] [->
ESPAGNOL ESPANOL] [COMMENT COMO]]
ECC :DICTIONNAIRE
[[MAISON CASA] [ESPAGNOL ESPANOL] [CO->
MMENT COMO]]

NOUVEAUMOT [TABLE MESA]
ECC :DICTIONNAIRE
[[MAISON CASA] [ESPAGNOL ESPANOL] [CO->
MMENT COMO] [TABLE MESA]]
```

MEMBREP opération

MEMBREP objet liste

MEMBREP (membre prédicat) retourne VRAI si objet est un élément de liste; sinon, retourne FAUX.

Exemples:

```
EC MEMBREP 3 [2 5 3 6]
VRAI

EC MEMBREP 3 [2 5 [3] 6]
FAUX

EC MEMBREP [2 5] [2 5 3 6]
FAUX

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

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

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

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

MOT opération

MOT mot1 mot2
(MOT mot1 mot2 mot3...)

Cette opération retourne un mot constitué de ses entrées. Si MOT a plus d'une entrée, il faut inclure la primitive et ses entrées entre parenthèses. La primitive MOT n'opère pas si une de ses entrées est une liste.

Exemples:

```
EC MOT "COU "LEUR
COULEUR

EC (MOT "APO "CALY "PSE)
APOCALYPSE

EC MOT "COU [LEUR]
MOT N'AIME PAS [LEUR] COMME ENTREE
```

La procédure SUFFIXE ajoute MUM à la fin de son entrée:

```
POUR SUFFIXE :MOT
RETOURNE MOT :MOT "MUM
FIN

EC SUFFIXE "MAXI
MAXIMUM
```

Le principe de la procédure SUFFIXE appliqué aux procédures LATIN et CUISINE traduit en latin de cuisine la phrase proposée:

```
POUR LATIN :PH
SI VIDEP :PH [RT []]
RT PH CUISINE PREMIER :PH LATIN SP :P->
H
FIN

POUR CUISINE :MOT
SI MEMBREP PREMIER :MOT [A E I O U] [->
RT MOT :MOT "MUM]
RT CUISINE MOT SP :MOT PREMIER :MOT
FIN

EC LATIN [PERSONNE N'A JAMAIS PARLE ->
LE LATIN DE CUISINE PARMI LES HUMAINS->
]
ERSONNEPMUM AN'MUM AMAISJMUM ARLEPMUM->
 ELMUM ATINLMUM EDMUM UISINECMUM ARMI->
PMUM ESLMUM UMAINSHMUM
```

MOTP opération

MOTP objet

MOTP (mot prédicat) retourne VRAI si objet est un mot; sinon,

retourne FAUX.

Exemples:

```
EC MOTP 3
VRAI

EC MOTP [3]
FAUX

EC MOTP "POUF
VRAI

EC MOTP [Ø GRE]
FAUX
```

MP opération

MP objet liste

MP (mets premier) retourne une nouvelle liste obtenue en ajoutant objet en première place de liste. Le tableau du début de ce chapitre compare MP aux autres opérations combinant mots et listes.

Exemple:

La procédure INVERSE place en ordre inverse les éléments de la liste d'entrée.

```
POUR INVERSE :LISTE
SI VIDEP :LISTE [RT [ ]]
RT MP DERNIER :LISTE INVERSE SD :LIST->
E
FIN

ECC INVERSE [[AV 2Ø] LC [DR 9Ø] [AV ->
2Ø] BC [RE 2Ø]]
[[RE 2Ø] BC [AV 2Ø] [DR 9Ø] LC [AV 2Ø->
]]
```

NOMBREP opération

NOMBREP objet

NOMBREP (nombre prédicat) retourne VRAI si objet est un nombre; sinon, retourne FAUX.

Exemples:

```
EC NOMBREP 3
VRAI
```

```
EC NOMBREP [3]
FAUX

EC NOMBREP "7PM
FAUX

EC NOMBREP "
FAUX

EC NOMBREP SP 3165.2
VRAI
```

PH opération

PH objet1 objet2
(PH objet1 objet2 objet3 ...)

PH (phrase) construit une liste composée des éléments contenus dans ses entrées. Se référer au début de ce chapitre où PH est comparée aux autres opérations qui combinent mots et listes.

Exemples:

```
ECC PH "PAPIER "CAHIER
[PAPIER CAHIER]

ECC PH "POMME [PECHE POIRE ABRICOT]
[POMME PECHE POIRE ABRICOT]

ECC PH [LE TEMPS PERDU] [NE REVIENT ->
GUERE]
[LE TEMPS PERDU NE REVIENT GUERE]
```

Si PH a plus de deux entrées, il faut placer la primitive et ses entrées entre parenthèses:

```
ECC (PH "POMME "PECHE "POIRE)
[POMME PECHE POIRE]

ECC PH "MONET []
[MONET]
```

La procédure suivante annonce une naissance:

```
POUR FAIREPART :PRENOM :NOM
EC [IL NOUS FAIT PLAISIR D'ANNONCER ->
LA NAISSANCE DE]
EC (PH :PRENOM :NOM)
EC [2.72 KG]
FIN

FAIREPART "GENEVIEVE "LABELLE
IL NOUS FAIT PLAISIR D'ANNONCER LA NA->
```

ISSANCE DE
GENEVIEVE LABELLE
2.72 KG

PREMIER opération

PREMIER objet

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

Exemples:

```
ECC PREMIER "PALAIS
P

ECC PREMIER [JOUJOU]
JOUJOU

ECC PREMIER [[LES TROIS] [CHAT CHIEN->
 RAT] [GRIFFE MORD RONGE]]
[LES TROIS]
```

La procédure ELEM retourne le :Nième élément de sa deuxième entrée.

```
POUR ELEM :N :OBJET
SI :N = 1 [RT PREMIER :OBJET]
RT ELEM :N - 1 SP :OBJET
FIN

EC ELEM 3 [CHAT CHIEN RAT PUCE]
RAT

EC ELEM 4 "ORANGE
N
```

SD opération

SD objet

SD (sauf dernier) retourne la liste ou le mot objet sauf son dernier élément. SD d'un mot vide ou d'une liste vide donne un message d'erreur.

Exemples:

```
ECC SD [ALBERT EINSTEIN]
[ALBERT]
```

```
ECC SD "HEURES
HEURE

ECC SD [HEURES]
[]
```

L'entrée de la procédure suivante doit être un verbe en <<er>>.

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

CONJUGUE "CHANTER
JE CHANTE
NOUS CHANTONS
```

SP opération

SP objet

SP (sauf premier) retourne la liste ou le mot objet sauf son premier élément. SP d'une liste vide ou d'un mot vide est une erreur.

Exemples:

```
ECC SP [ALBERT EINSTEIN]
[EINSTEIN]

ECC SP "TROIS
ROIS

ECC SP [TROIS]
[]                              Liste vide.

ECC SP [LES TROIS]
[TROIS]

ECC SP [[LES DEUX] [CHIEN CHAT] [JAP->
PE MIAULE]]
[[CHIEN CHAT] [JAPPE MIAULE]]

EC SP "
SP N'AIME PAS  COMME ENTREE

EC SP []
SP N'AIME PAS [ ] COMME ENTREE
```

La procédure suivante enlève un élément d'un mot ou d'une liste:

```
POUR TRIANGLE :OBJET
SI VIDEP :OBJET [STOP]
EC :OBJET
TRIANGLE SP :OBJET
FIN

TRIANGLE "TROIS
TROIS
ROIS
OIS
IS
S

TRIANGLE [A PAS DE CHAT]
A PAS DE CHAT
PAS DE CHAT
DE CHAT
CHAT
```

VIDEP opération

VIDEP objet

VIDEP (vide prédicat) retourne VRAI si objet est le mot vide ou la liste vide; sinon, retourne FAUX.

Exemples:

```
EC VIDEP "
VRAI

EC VIDEP []
VRAI

EC VIDEP Ø
FAUX

EC VIDEP SP "CHAPEAU
FAUX

EC VIDEP SD "U
VRAI

EC VIDEP SP [CHAPEAU]
VRAI
```

La procédure CRIS relie des animaux à leurs cris respectifs:

```
POUR CRIS :ANIMAUX :BRUITS
SI OU VIDEP :BRUITS VIDEP :ANIMAUX [E->
C [VOILA, C'EST TOUT] STOP]
EC PH PREMIER :ANIMAUX PREMIER :BRUIT->
S
```

```
CRIS SP :ANIMAUX SP :BRUITS
FIN

CRIS [CHIENS OISEAUX COCHONS] [ABOIE->
NT PEPIENT GROGNENT]
CHIENS ABOIENT
OISEAUX PEPIENT
COCHONS GROGNENT
VOILA, C'EST TOUT
```

= (symbole d'égalité) opération infixe

objet1 = objet2

Cette opération retourne VRAI si objet1 et objet2 sont des nombres égaux, des mots identiques, ou des listes identiques; sinon, elle retourne FAUX. Le symbole = équivaut à l'opération préfixe EGALP (égal prédicat).

Exemples:

```
EC 3 = PREMIER "3.1416
VRAI

EC [AMOUR] = [INDIFFERENCE]
FAUX

EC 7. = 7                    Un nombre décimal équivaut à
VRAI                         l'entier correspondant.

EC " = [ ]                   Le mot vide et la liste vide
FAUX                         ne sont pas identiques.
```

Chapitre 3

Les variables

Un mot Logo peut s'utiliser comme variable; une variable est un contenant qui renferme un objet Logo. Cet objet est appelé la valeur du mot. Ceci se réalise soit en utilisant RELIE, soit par les entrées d'une procédure.

CHOSE opération

CHOSE nom

Cette opération révèle l'objet contenu dans la boîte nom, c'est-à-dire la valeur de la variable nom. CHOSE "QQCH est équivalent à :QQCH. Le contenu de la variable peut être créé en utilisant la commande RELIE ou en définissant une procédure avec entrées.

Exemples:

```
RELIE "GAGNANT "ORDINATEUR
RELIE "ORDINATEUR [1ØØ POINTS]
EC CHOSE "GAGNANT
ORDINATEUR

EC :GAGNANT
ORDINATEUR

EC CHOSE :GAGNANT
1ØØ POINTS
```

La procédure suivante utilise un incrément, c'est-à-dire une valeur constante qu'elle ajoute à celle d'une variable:

```
POUR INC :X
SI NON NOMP :X [STOP]
SI NOMBREP CHOSE :X [RELIE :X 1 + CHO—>
SE :X]
FIN
```

L'utilisation de RELIE :X plutôt que RELIE "X est à noter. En effet, ce n'est pas à X que doit s'appliquer l'incrément, car la valeur de X n'est pas un nombre mais le nom d'une autre variable; c'est donc la valeur de cette dernière que l'incrément doit augmenter.

```
RELIE "TOTAL 7
EC :TOTAL
7

INC "TOTAL
EC :TOTAL
8

INC "TOTAL
EC :TOTAL
```

On trouvera d'autres exemples sous la rubrique RELIE.

NOMP opération

NOMP nom

NOMP (nom prédicat) répond VRAI si nom existe, c'est-à-dire si nom a une valeur; sinon, répond FAUX.

Exemples:

```
EC :ANIMAL
ANIMAL N'EST PAS RELIEE

EC NOMP "ANIMAL
FAUX

RELIE "ANIMAL "RHINOCEROS
EC NOMP "ANIMAL
VRAI

EC :ANIMAL
RHINOCEROS
```

La procédure INC, employée dans l'exemple de la rubrique CHOSE, montre une utilisation de NOMP.

RELIE commande

RELIE nom objet

Cette commande met l'objet dans le contenant de nom, c'est-à-dire donne à la variable nom la valeur d'objet, ou plus simplement, relie l'un à l'autre. Une fois que la variable est créée, l'opération CHOSE nom y donne accès. L'abréviation :nom équivaut à CHOSE "nom. Les deux points (:) signifient <<la chose nommée>>.

Exemples:

```
RELIE "NATIONS [CANADA SUISSE JAPON BRE->
SIL ISRAEL]
EC :NATIONS
CANADA SUISSE JAPON BRESIL ISRAEL

EC "NATIONS
NATIONS

EC CHOSE "NATIONS
CANADA SUISSE JAPON BRESIL ISRAEL
```

```
RELIE "SUISSE [GENEVE]
EC :SUISSE
GENEVE

EC CHOSE PREMIER SP :NATIONS
GENEVE
```

PREMIER SP :NATIONS est le second nom dans la liste des nations, en l'occurrence SUISSE, et la valeur de CHOSE "SUISSE est GENEVE.

```
RELIE "CANADA [OTTAWA]
EC PREMIER :NATIONS
CANADA

EC CHOSE PREMIER :NATIONS
OTTAWA
```

La procédure suivante, CAPITALE, demande le nom des capitales des pays mentionnés.

```
POUR CAPITALE :NATIONS
SI VIDEP :NATIONS [STOP]
RELIE "PAYS PREMIER :NATIONS
EC PH [LA CAPITALE DE] :PAYS
RELIE "REPONSE LISL
SI :REPONSE = CHOSE :PAYS [EC [EXACT!->
]] [EC [AH! DEPUIS QUAND?]]
CAPITALE SP :NATIONS
FIN

CAPITALE :NATIONS
LA CAPITALE DE CANADA
OTTAWA
EXACT!
LA CAPITALE DE SUISSE
MONTREAL
AH! DEPUIS QUAND?
```

Chapitre 4

Opérations arithmétiques

Logo reconnaît des nombres entiers et des nombres décimaux:

3 est un entier;
3.14 est un nombre décimal.

Logo fournit des primitives pour additionner, soustraire, multiplier et diviser les nombres, pour trouver les sinus, cosinus et racines carrées, et pour vérifier si un nombre est égal, inférieur ou supérieur à un autre nombre.

Certaines opérations arithmétiques (ENT, HASARD, RESTE, ARRONDIS) retournent toujours un entier tandis que d'autres retournent soit un entier, soit un nombre décimal.

Les nombres décimaux à plus de 6 chiffres sont convertis à la forme "exponentielle" (notation scientifique). Par exemple:

2E6 signifie 2 multiplié par 10^6, soit 2,000,000;
2.59E-2 signifie 2.59 multiplié par 10^{-2}, soit 0.0259.

Les valeurs des exposants sont comprises entre -99 et 97. Logo arrondit les nombres décimaux qui contiennent plus de 9 chiffres. Par exemple: 2718281828459.045 devient 2.71828182E+12.

L'addition, la soustraction, la multiplication et la division ont une forme infixe. Dans ce cas l'infixe se place entre les entrées sur lesquelles il opère. L'addition et la multiplication disposent aussi d'une forme préfixe en tant qu'opérations Logo à deux entrées ou plus. Par exemple, les deux expressions suivantes sont équivalentes:

```
2 + 1
SOMME 2 1
```

En plus des primitives listées ci-après, la primitive EGALP, fréquemment utilisée en relation avec des opérations arithmétiques, est décrite au chapitre 2 (Mots et listes). L'opération infixe =, symbole d'égalité, en est l'équivalent.

ARRONDIS opération

ARRONDIS n

Cette opération retourne n arrondi à l'entier le plus proche. Comparer avec les exemples de ENT.

Exemples:

```
EC ARRONDIS 5.2129
5

EC ARRONDIS 5.5129
```

6

ENT opère différemment:

```
EC ENT 5.5129
5

EC ARRONDIS .5
1

EC ARRONDIS -5.8
-6

EC ARRONDIS -12.3
-12
```

COS opération

COS n

COS (cosinus) donne le cosinus de n degrés. Il y a erreur si n est plus grand que 9999.9999 ou plus petit que -9999.9999.

Exemples:

```
EC COS 45
Ø.7Ø714

EC COS 3Ø
0.866Ø5
```

Voici une définition de la fonction tangente:

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

EC TG 45
1
```

ENT opération

ENT n

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

Exemples:

```
EC ENT 5.2129
5
```

```
EC ENT 5.5129
5

EC ENT 5
5

EC ENT -5.8
-5

EC ENT -12.3
-12
```

La procédure suivante indique si un nombre est entier ou non:

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

RT → retourne

```
EC ENTP 17
VRAI

EC ENTP 1ØØ / 8
FAUX

EC ENTP "UN
PAS UN NOMBRE

EC ENTP RC 5Ø
FAUX
```

HASARD opération

HASARD n

Cette 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
RT 1 + HASARD 6
FIN

EC DE6
3

EC DE6
5
```

```
EC DE6
3
```

Note: Les sorties de D6 données en exemple sont des nombres aléatoires, précisément à cause du HASARD.

PRODUIT opération

```
PRODUIT a b
(PRODUIT a b c ...)
```

Cette opération donne le produit des entrées a, b, ... La forme infixe * équivaut à PRODUIT. Si PRODUIT a plus de deux entrées, il faut placer la primitive et ses entrées entre parenthèses.

Exemples:

```
EC PRODUIT 6 2
12

EC (PRODUIT 2 3 4)
24

EC PRODUIT 2.5 4
1Ø

EC PRODUIT 2.5 2.5
6.25
```

RC opération

```
RC n
```

RC (racine carrée) donne pour résultat la racine carrée de n. Il y a erreur si n est négatif.

Exemples:

```
EC RC 25
5

EC RC 259
16.Ø93477
```

La procédure suivante retourne la distance entre la position actuelle de la Tortue et son point d'origine.

```
POUR DIST.ORIGINE
RT ARRONDIS RC SOMME XCOR * XCOR YCOR->
 * YCOR
FIN
```

```
AV 5Ø
EC DIST.ORIGINE
5Ø
```

La procédure DISTANCE prend comme entrées 2 positions et retourne la distance entre elles:

```
POUR DISTANCE :POS1 :POS2
RELIE "X (PREMIER :POS1) - PREMIER :P—>
OS2
RELIE "Y (DERNIER :POS1) - DERNIER :P—>
OS2
RT RC :X * :X + :Y * :Y
FIN

EC DISTANCE [-7Ø 1Ø] [5Ø 6Ø]
129.9999
```

REHASARD commande

```
REHASARD
```

REHASARD (reproduit hasard) rend le résultat de HASARD reproductible: après avoir effectué REHASARD, les appels à HASARD retourneront chaque fois la même séquence de résultats aléatoires. L'entrée de HASARD doit être la même qu'à la première exécution de REHASARD.

Exemples:

```
REPETE 4 [EC HASARD 1Ø]
5
2
8
4

REHASARD REPETE 4 [EC HASARD 1Ø]
8
2
3
2

REHASARD REPETE 4 [EC HASARD 1Ø]
8
2
3
2
```

RESTE opération

```
RESTE a b
```

Cette opération retourne le reste de la division de a par b. Il

y a erreur si la valeur de b est Ø.

```
EC RESTE 13 5
3
```

13 divisé par 5 est 2 et le reste est 3.

```
EC RESTE 13 15
13

EC RESTE -13 5
-3
```

La procédure suivante détermine si un nombre est pair:

```
POUR PAIRP :NOMBRE
RT Ø = RESTE :NOMBRE 2
FIN

EC PAIRP 5
FAUX

EC PAIRP 12462
VRAI
```

SIN opération

SIN n

SIN (sinus) exprime la valeur du sinus de n degrés. Voir également COS (cosinus).

Exemple:

```
EC SIN 45
Ø.7Ø714
```

SOMME opération

SOMME a b
(SOMME a b c ...)

Opération qui effectue l'addition de ses entrées, la primitive SOMME équivaut au symbole +, opération infixe. Si SOMME a plus de deux entrées, les entrées et la primitive doivent être mises entre parenthèses.

Exemples:

```
EC SOMME 5 2
7
```

```
EC (SOMME 1 3 2 -1)
5

EC SOMME 2.3 2.561
4.861
```

+ (symbole d'addition) opération infixe

a + b

Cette opération infixe additionne ses entrées. Elle équivaut à SOMME, opération préfixe.

Exemples:

```
EC 5 + 2
7

EC 1 + 3 + 2 + 1
7

EC 2.54 + 12.3
14.84
```

- (symbole de soustraction) opération infixe

a - b

Cette opération infixe retourne le résultat obtenu en soustrayant b de a. Elle peut être utilisée comme signe négatif d'un nombre.

Exemples:

```
EC 7 - 1
6

EC 7-1
6

EC PRODUIT 7 -1
-7

EC -XCOR                    Ce nombre varie selon la position
-5Ø                         actuelle de la Tortue.

EC - 3
-3

EC -3 - -2
-1
```

Il faut noter qu'il peut y avoir confusion entre un signe négatif à une entrée et le symbole de soustraction à deux entrées. Logo résout l'ambiguïté ainsi:

```
EC 3 * -4                    3 fois le nombre négatif 4
-12

EC 3 + 4 - 5                 3 plus 4 moins 5
2
```

S'il y a un espace avant le <<->> suivi immédiatement d'un nombre, Logo interprète ce dernier comme un nombre négatif. Ainsi, 7 - 1 donne 6, mais 7 -1 ne signifie qu'une paire de nombres pour Logo: 7 et -1.

La procédure ABS retourne la valeur absolue de son entrée:

```
POUR ABS :NOMB
RT SI :NOMB < Ø [-:NOMB] [:NOMB]
FIN

EC ABS -35
35

EC ABS 35
35
```

* (symbole de multiplication) opération infixe

a * b

Cette opération infixe retourne le produit de ses entrées. Elle équivaut à la forme préfixe PRODUIT.

Exemples:

```
EC 6 * 2
12

EC 2 + 3 * 4
14

EC 1.3 * -1.3
-1.69

EC 48 * (.3 + .2)
24
```

La procédure FACTORIELLE retourne la factorielle de son entrée. Par exemple, FACTORIELLE 5 donne le résultat de 5 * 4 * 3 * 2 * 1 (12Ø).

```
POUR FACTORIELLE :N
```

```
SI :N = Ø [RT 1]
RT :N * FACTORIELLE :N - 1
FIN

EC FACTORIELLE 4
24

EC FACTORIELLE 1
1
```

/ (symbole de division) opération infixe

a / b

Opération infixe qui retourne a divisé par b.

Exemples:

```
EC 6 / 3
2

EC 8 / 3
2.66666666

EC 2.5 / -3.8
-Ø.6578947368

EC Ø / 7
Ø
```

Il y a erreur si b a comme valeur Ø:

```
EC 7 / Ø
/ N'AIME PAS Ø COMME ENTREE
```

< (symbole de plus petit que) opération infixe

a < b

Cette opération infixe retourne VRAI si a est inférieur à b; sinon, elle retourne FAUX.

Exemples:

```
EC 2 < 3
VRAI

EC -7 < -1Ø
FAUX
```

= (symbole d'égalité) opération infixe

a = b

La réponse de cette opération infixe est VRAI si a et b sont des nombres égaux, des mots identiques, ou des listes identiques. Dans le cas contraire, la réponse est FAUX. La primitive = équivaut à EGALP (égal prédicat), opération préfixe.

Exemples:

```
EC 1ØØ = 5Ø * 2
VRAI

EC 3 = PREMIER "3.1416
VRAI

EC 7. = 7          Un nombre décimal est équivalent à l'entier
VRAI               correspondant.

EC " = []          Le mot vide n'est pas la même
FAUX               chose que la liste vide.
```

> (symbole de plus grand que) opération infixe

a > b

Cette opération infixe retourne VRAI si a est supérieur à b; sinon, elle retourne FAUX.

Exemples:

```
EC 4 > 3
VRAI
```

La procédure ENTRE retourne VRAI si le nombre de la première entrée est plus grand que celui de la deuxième entrée, et plus petit que celui de la troisième entrée.

```
POUR ENTRE :N :BAS :HAUT
RT ET :N > :BAS :HAUT > :N
FIN

EC ENTRE 15 Ø 16
VRAI

EC ENTRE -5 -2 5
FAUX
```

Chapitre 5

Définition et édition de procédures

Il existe deux façons de définir des procédures. La première utilise POUR, la seconde emploie EDITE. La primitive POUR permet de définir une procédure au niveau supérieur sans perdre l'écran graphique. L'emploi de la primitive EDITE fait disparaître l'écran graphique mais procure un éditeur de texte interactif où tout changement apporté est affiché à l'écran. Il est ainsi possible d'éditer plus d'une procédure à la fois. Ce moyen est donc plus flexible lorsqu'on doit effectuer des modifications. Même si l'éditeur de texte est largement utilisé, chaque méthode a ses avantages et il appartient à l'utilisateur de décider laquelle il préfère.

L'éditeur Logo d'ATARI

Fonctionnement de l'éditeur

La commande EDITE provoque l'apparition de l'éditeur de texte à l'écran. Par exemple:

```
EDITE "POLY

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

EDITEUR ATARI LOGO

Il n'y a pas de symbole d'invite mais le curseur indique la place où s'inscrit le caractère tapé. Le curseur se déplace n'importe où dans le texte à l'aide des touches de contrôle. Des caractères peuvent être effacés ou insérés en utilisant les touches appropriées décrites dans ce chapitre.

Il arrive qu'il y ait plus de caractères dans une ligne de texte que ne peut en contenir la largeur d'une ligne d'écran (37). Quand la fin de celle-ci est atteinte, il s'agit simplement de continuer à taper sans presser la touche RETOUR. Une flèche (—>) apparaît alors à la fin de la ligne d'écran, à la place du 38^{e} caractère, et le curseur est reporté à la ligne suivante. Logo agit de même hors de l'éditeur. Il est possible de composer des lignes de texte de n'importe quelle longueur à l'intérieur de l'éditeur. En dehors de ce dernier, la longueur maximale d'une ligne Logo est de 11Ø caractères.

Voici comment une longue ligne de texte apparaît à l'écran:

```
POUR IMMESSAGE :AMI
EC PH :AMI [, JE VAIS ECRIRE UN TRES ->
```

```
LONG MESSAGE POUR TOI]
FIN
```

L'éditeur possède un tampon réserve capable de conserver une ligne de texte enlevée à l'aide de la combinaison des touches HAUT et EFF/ARR. L'emploi de la touche CTRL combinée à la touche de caractère Y réinsère cette ligne de texte à l'endroit désiré après que le curseur y ait été ramené. Le tampon réserve peut contenir jusqu'à 11Ø caractères.

Le texte édité fait partie d'une mémoire tampon d'édition. Ce tampon a une capacité de 384Ø caractères.

Les touches fléchées, CTRL ou HAUT et quelques touches de caractère combinées à la touche CTRL ont des significations spéciales facilitant l'édition.

Actions d'édition

Les touches d'édition du tableau suivant fonctionnent à l'intérieur de l'éditeur. Celles qui sont précédées d'un astérisque fonctionnent à la fois à l'intérieur et à l'extérieur de l'éditeur Logo d'ATARI.

Pour déplacer le curseur

* CTRL ->	Avance le curseur d'un espace vers la droite.
* CTRL <-	Recule le curseur d'un espace vers la gauche.
CTRL ↓	Descend le curseur à la ligne suivante.
CTRL ↑	Remonte le curseur à la ligne précédente.
* CTRL A	Ramène le curseur au début de la ligne courante.
* CTRL E	Renvoie le curseur à la fin de la ligne courante.
CTRL X	Reporte le curseur au début de l'éditeur.
CTRL Z	Envoie le curseur à la fin de l'éditeur.

Logo émet un signal sonore à chaque fois qu'on tente d'envoyer le curseur où il n'y a plus de texte.

Pour insérer et effacer

* CTRL EFF/ARR	Efface le caractère que recouvre le curseur. Comparer avec EFF/ARR.
CTRL INSERE	Ouvre une nouvelle ligne à la position du curseur sans le déplacer.
* CTRL VIDE	Efface le texte à partir de la position du curseur jusqu'à la fin de la ligne courante. Ce texte est placé dans le tampon réserve qui peut contenir jusqu'à 11Ø caractères.
* CTRL Y	Insère le texte conservé dans le tampon réserve.
* EFF/ARR	Efface le caractère à gauche du curseur.
* HAUT EFF/ARR	Equivaut à CTRL VIDE.
HAUT INSERE	Equivaut à CTRL INSERE.
* RETOUR	Ouvre une nouvelle ligne à la position actuelle du curseur et renvoie ce dernier au début de cette nouvelle ligne.

Pour changer de page à l'écran

CTRL 1	Au niveau supérieur, arrête le déroulement des pages de l'écran jusqu'à ce que la combinaison des touches CTRL et 1 soit pressée à nouveau.
CTRL V	Amène la page suivante à l'écran dans l'éditeur.
CTRL W	Déroule la page précédente dans l'éditeur.

Pour sortir de l'éditeur

QUITTE	Façon régulière de quitter l'éditeur. Lorsque cette touche est employée, Logo lit chaque ligne de la mémoire tampon d'édition comme si elle avait été tapée hors de l'éditeur. Logo fournit le mot FIN si on quitte l'éditeur en omettant de le taper quand la définition de la procédure est terminée.

Il est possible de définir plus d'une procédure à la fois dans l'éditeur à la condition d'écrire FIN en terminant chacune d'elles.

Au moment de quitter l'éditeur, si la mémoire tampon d'édition contient des instructions qui ne font pas partie de la définition d'une procédure (comprises entre POUR et FIN), Logo les ramène hors de l'éditeur comme si elles y avaient été tapées au niveau supérieur. Logo n'effectuera cependant aucune commande graphique ou d'édition.

ARRET — Arrête l'édition. On emploie cette touche pour stopper les modifications en cours et sortir de l'éditeur sans qu'il en soit tenu compte. La procédure demeure ainsi telle que définie préalablement.

EDITE, ED commande

EDITE nom
EDITE listenom

Cette commande démarre l'éditeur Logo d'ATARI. Selon l'entrée donnée à EDITE, l'éditeur commence avec la ou les définitions de la ou des procédures contenues dans la mémoire tampon d'édition. L'entrée de EDITE peut être un nom de procédure ou une liste de noms de procédures. Dans ce dernier cas, la définition de chacune des procédures de la liste est amenée dans l'éditeur.

Si le nom de la procédure n'a pas encore été défini, la mémoire tampon d'édition ne contient que la ligne du titre: POUR nom. Si aucune entrée n'est donnée à EDITE, la mémoire tampon d'édition ne contient que les procédures préalablement définies au moyen de l'éditeur. La mémoire tampon d'édition est vide lorsque l'éditeur est utilisé pour la première fois.

La touche QUITTE doit être pressée pour compléter une définition et sortir de l'éditeur. Logo interprète chaque ligne comme si elle avait été tapée hors de l'éditeur. Si la capacité maximale de la mémoire tampon d'édition est atteinte alors qu'une définition de procédure est en cours dans l'éditeur, Logo y met un terme en insérant le mot FIN.

EDNS commande

EDNS

EDNS (édite noms) démarre l'éditeur Logo qui affiche tous les noms définis et leurs valeurs, qui peuvent alors être édités. Au sortir de l'éditeur les commandes RELIE sont exécutées; ainsi toutes les variables et les valeurs qui ont été changées dans l'éditeur sont changées en Logo.

Exemple:

On tape:

```
EDNS
```

L'écran apparaît ainsi:

```
RELIE "ANIMAL "RHINOCEROS
RELIE "VITESSE 55
RELIE "AVION [BALLON HELICOPTERE]
```

On édite les noms de la façon suivante et on presse la touche QUITTE pour sortir de l'éditeur:

```
RELIE "ANIMAL "LION
RELIE "VITESSE 55
RELIE "AVION [BALLON HELICOPTERE NAVE->
TTE]
```

Puis:

```
IMNS
RELIE "ANIMAL "LION
RELIE "VITESSE 55
RELIE "AVION [BALLON HELICOPTERE NAVE->
TTE]
```

FIN mot spécial

```
FIN
```

Lorsqu'on utilise la commande POUR afin de définir une procédure, il est indispensable d'employer le mot FIN pour indiquer à Logo que la définition est complétée. Ce mot doit figurer seul comme dernière ligne de la procédure. Il doit aussi être tapé à la fin de chaque procédure pour les séparer les unes des autres lorsqu'on en définit plusieurs à la fois dans l'éditeur.

POUR commande

POUR nom entréel entrée2 ...

```
?POUR SALUTATION
>EC [BONJOUR]
```

```
>FIN
SALUTATION DEFINIE
?
```

```
?POUR CARRE :COTE
>AV :COTE
>DR 9Ø
>AV :COTE
>DR 9Ø
>AV :COTE
>DR 9Ø
>AV :COTE
>DR 9Ø
>FIN
CARRE DEFINIE
?
```

POUR indique à Logo qu'on est à définir une procédure appelée nom et ses entrées, s'il y a lieu. Il n'est pas nécessaire de citer le nom de la procédure en le faisant précéder d'un guillemet, POUR le considère cité implicitement. Le symbole d'invite change de << ? >> à << > >>, ce qui indique qu'on est à définir une procédure. A ce moment, Logo n'effectue pas les instructions tapées, il les inscrit simplement dans la définition de la procédure.

Pour compléter la procédure et retourner Logo au niveau supérieur, on doit taper le mot FIN, seul, comme dernière ligne de la procédure.

La touche ARRET sert à interrompre la définition d'une procédure commencée à l'aide de POUR. Si la définition d'une procédure est complétée, elle ne peut être modifiée en utilisant POUR. Il faut l'effacer au complet par la commande EF (efface) ou l'amener dans l'éditeur par la commande EDITE afin d'y apporter les changements voulus.

Chapitre 6

Contrôle d'exécution et instructions conditionnelles

Logo interprète les définitions de procédures ligne par ligne. A l'intérieur d'une superprocédure, si Logo rencontre une sous-procédure, il lit d'abord les lignes de cette dernière avant de continuer la lecture de la superprocédure. Le contrôle d'exécution est l'ordre dans lequel Logo exécute les instructions. Cet ordre peut être modifié de différentes manières:

— Par une instruction conditionnelle:
<<Si cette condition s'avère vraie, fais telle chose, sinon fais telle autre chose>>;

— Par une instruction d'itération:
<<Recommence telle liste d'instructions une ou plusieurs fois>>;

— Par une instruction d'arrêt:
<<Arrête cette procédure avant qu'elle n'atteigne le mot FIN>>.

L'instruction conditionnelle permet à Logo d'effectuer différentes instructions si une certaine condition est rencontrée. Un prédicat Logo, opération qui retourne VRAI ou FAUX, crée cette condition qui doit être la première entrée de SI.

L'instruction d'itération suscite la répétition d'instructions en utilisant la primitive REPETE ou en créant une procédure récursive. On trouvera plusieurs exemples de telles procédures tout au long de ce manuel. Consulter la description de la primitive EXECUTE pour trouver des exemples complexes de procédures récursives.

L'instruction d'arrêt provoque une interruption de la procédure avant qu'elle ne s'exécute jusqu'au mot FIN. Les commandes STOP et RETOURNE servent à cette interruption. Le contrôle d'exécution est alors transmis à la procédure appelante ou au niveau supérieur. Tel que décrit dans la grammaire Logo, RETOURNE peut fournir des informations à la procédure appelante. Les commandes STOP et RETOURNE n'agissent qu'à l'intérieur de la procédure où elles figurent.

Le QUAND Diablotin est un moyen complètement différent d'influencer le contrôle d'exécution. Il s'agit d'une condition globale qui n'a besoin d'être établie qu'une fois. Lorsqu'elle est rencontrée dans n'importe quelle procédure, ou au niveau supérieur, une série d'instructions s'exécute.

Le QUAND Diablotin est semblable à un lutin malicieux tapi dans le monde Logo et surveillant constamment qu'une collision ou qu'une éventualité se produise. Il réagit alors immédiatement en commandant à Logo d'exécuter une liste d'instructions. Ceci fait, il regagne son poste d'observation.

La primitive QUAND prépare les conditions d'observation du Diablotin. Les collisions et les éventualités pouvant être

vérifiées par QUAND apparaissent dans le tableau qui suit. Les primitives COLTC (collision tortue crayon) et COLTT (collision tortue tortue) sont utiles pour mémoriser le numéro correspondant à une collision ou à une éventualité. On en donne des exemples à l'appendice B du manuel Introduction à la programmation via le graphique Tortue.

CONDP (condition prédicat) est une autre primitive capable de vérifier l'occurrence de collisions ou d'éventualités. A l'inverse de QUAND, cette primitive ne peut opérer qu'au moment où Logo lit la ligne qui contient une telle éventualité.

Tableau des collisions et des éventualités

Numéro de code		Entrées		
Collisions	Eventualités	Numéro de Tortue	Numéro de crayon	Description des éventualités
Ø		Ø	Ø	
1		Ø	1	
2		Ø	2	
	3			Le bouton du levier est pressé
4		1	Ø	
5		1	1	
6		1	2	
	7			Une fois par seconde
8		2	Ø	
9		2	1	
1Ø		2	2	
11				Inutilisé
12		3	Ø	
13		3	1	
14		3	2	
	15			La position du levier est changée

Numéro de collision	Numéro de Tortue	Numéro de Tortue
16	3	Ø
17	3	1
18	3	2
19	Ø	1
2Ø	Ø	2
21	1	2

Chaque numéro du tableau symbolise une collision ou une éventualité. Par exemple, la collision numéro Ø signifie une collision entre la Tortue Ø et une ligne tracée par le crayon numéro Ø. L'éventualité numéro 3 se produit si le bouton d'une manette est pressé.

Note: Il est préférable de travailler à l'écran graphique (ECRANG) lorsqu'on utilise les QUAND Diablotins 2, 6, 1Ø et 14, car dans l'écran partagé (ECRANP), les Tortues peuvent entrer en collision avec le texte.

ATTENDS commande

ATTENDS n

Cette commande avertit Logo de suspendre l'exécution d'une procédure pendant n 6Øièmes de seconde.

Exemple:

La procédure AVLENT fait avancer la Tortue très lentement:

```
POUR AVLENT :DIST
REPETE :DIST [AV 1 ATTENDS 1]
FIN

AVLENT 8Ø
VE
REPETE 4 [AVLENT 8Ø DR 9Ø]
```

COLTC opération

COLTC numérotortue numérocrayon

COLTC (collision tortue crayon) retourne un numéro symbolisant une collision entre une Tortue (numérotortue) et le tracé d'un crayon (numérocrayon). Voir le tableau des collisions et des éventualités au début de ce chapitre. La commande COLTC peut être utilisée comme entrée de QUAND et de CONDP.

Exemple:

```
EC COLTC 1 Ø
4
```

COLTT opération

COLTT numérotortue1 numérotortue2

COLTT (collision tortue tortue) retourne le numéro symbolisant une collision entre numérotortue1 et numérotortue2. Voir le tableau des collisions et des éventualités au début de ce chapitre. L'opération COLTT peut être utilisée comme entrée de QUAND ou de CONDP.

Exemple:

```
EC COLTT 2 3
18
```

CONDP opération

CONDP numérocond

CONDP (condition prédicat) retourne VRAI si la collision ou l'éventualité spécifiée par le numérocond se produit au moment même où CONDP est exécutée. Sinon, cette opération retourne FAUX. L'entrée de CONDP doit être un entier de Ø à 21 indiquant quelle condition ou éventualité on veut vérifier (consulter le tableau de la page 31). La primitive est surtout utile pour ne vérifier qu'une fois l'occurrence d'une collision ou d'un éventualité. Comparer cette opération à la commande QUAND.

Exemple:

Dans l'exemple suivant la forme numéro 1 est une boîte pleine servant de cible. CONDP vérifie si elle a été atteinte en cherchant si une collision est survenue entre la Tortue numéro Ø et la Tortue numéro 1.

```
POUR TIRE
PREPARE
EC [DE COMBIEN DE DEGRES VOULEZ\-VOUS->
 TOURNER A DROITE?]
DR PREMIER LISL
EC [DE COMBIEN DE PAS VOULEZ\-VOUS AV->
ANCER?]
AV PREMIER LISL
SI CONDP 19 [EC [BRAVO!]] [EC [ZUT!]]->

FIN

POUR PREPARE
DESIGNE [Ø 1] VE MT
DESIGNE Ø LC
DR HASARD 36Ø
AV HASARD 8Ø
FCAP Ø BC
FFOR 1
DESIGNE 1
FIN

TIRE
DE COMBIEN DE DEGRES VOULEZ-VOUS TOUR->
NER A DROITE?
45
DE COMBIEN DE PAS VOULEZ-VOUS AVANCER->
?
5Ø
```

EXECUTE commande ou opération

EXECUTE listeinstruction

Utilisée en tant que commande, cette primitive effectue la liste d'instructions indiquées comme si elles étaient tapées directement. En tant qu'opération, EXECUTE retourne ce que la listeinstruction retourne.

Exemples:

La procédure suivante simule une calculatrice:

```
POUR CALCULATRICE
EC EXECUTE LISL
EC [ ]
CALCULATRICE
FIN

CALCULATRICE
2 + 3
5

17.5 * 3
52.5

42 = 8 * 7
FAUX

RESTE 12 5
2
```

On doit presser la touche ARRET pour stopper l'exécution de la procédure CALCULATRICE.

La procédure TANTQUE accomplit une liste d'instructions tant qu'une condition est considérée comme vraie:

```
POUR TANTQUE :COND :LISTEINST
SI NON EXECUTE :COND [STOP]
EXECUTE :LISTEINST
TANTQUE :COND :LISTEINST
FIN
```

```
DR 9Ø
TANTQUE [XCOR < 1ØØ] [AV 25 EC POS]
25 Ø
5Ø Ø
75 Ø
1ØØ Ø
```

La procédure qui suit applique une commande à chaque élément d'une liste à tour de rôle:

```
POUR APPLIQUE :COM :LISTE
SI VIDEP :LISTE [STOP]
EXECUTE LISTE :COM MOT "" PREMIER :LI->
STE
APPLIQUE :COM SP :LISTE
FIN

POUR CARRE :COTE
REPETE 4 [AV :COTE DR 9Ø]
FIN

APPLIQUE "CARRE [1Ø 2Ø 4Ø 8Ø]
```

```
RELIE "MARITIMES [NB NE IPE TN]
APPLIQUE "EC :MARITIMES
NB
NE
IPE
TN
```

La procédure suivante, INFINI, répète sans cesse son entrée à moins qu'elle ne rencontre une erreur ou qu'elle ne soit interrompue en pressant la touche ARRET.

```
POUR INFINI :LISTEINST
EXECUTE :LISTEINST
INFINI :LISTEINST
FIN
```

La commande INFINI [AV 1 DR 1] demande à la Tortue de dessiner un cercle.

```
INFINI [AV 1 DR 1]
```

La commande INFINI [EC EXECUTE LISL EC [ ]] équivaut à la procédure CALCULATRICE définie plus haut.

EXECUTE LISL effectue n'importe quelle commande ou opération tapée par l'utilisateur.

EC EXECUTE LISL écrit la sortie de toute expression tapée par l'utilisateur.

QUAND commande

QUAND numérocond listeinstruction
QUAND numérocond [ ]

Cette commande met en place un QUAND Diablotin capable de détecter une collision ou une éventualité numérocond. Voir le tableau au début de ce chapitre. Numérocond doit être un entier de Ø à 21 symbolisant une collision ou une éventualité. Lorsque l'une ou l'autre se produisent, la listeinstruction est effectuée. Si la listeinstruction contient des commandes Tortue, elles seront exécutées par la ou les Tortues désignées.

La commande QUAND doit être donnée alors que l'écran est en mode partagé ou en mode graphique.

L'effet de QUAND est global, c'est-à-dire qu'il n'est nécessaire de donner cette commande qu'une seule fois. Voir IMD (imprime diablotin) et IMDS (imprime diablotins) au chapitre 9 qui décrivent comment vérifier quels sont les Diablotins à l'oeuvre.

QUAND numérocond [ ]

Comme presque tous les numérocond réfèrent à une commande de graphique, il peut s'avérer impossible de travailler à l'écran de texte tant qu'un QUAND Diablotin demeure à l'affût. Il y a deux façons de le congédier. La façon la plus simple est d'employer la commande VE (vide écran) mais le graphique qui occupe l'écran est alors perdu. L'autre méthode consiste à donner la commande QUAND numérocond [ ] puisqu'un Diablotin qui ne trouve pas de travail reste inactif. Par exemple, si l'on désire congédier les QUAND Diablotins Ø et 4, il faut taper:

```
QUAND Ø [ ]
QUAND 4 [ ]
```

IMDS permet de vérifier si ces Diablotins guettent encore:

IMDS

Aucun Diablotin ne fait le guet.

Il est possible de donner plus d'une commande QUAND à la fois, mais les Diablotins ne peuvent être à l'oeuvre simultanément.

La rapidité avec laquelle ils détectent une collision ou une éventualité dépend de leur force. Le QUAND Diablotin Ø est le plus fort et partant le plus rapide. Le QUAND Diablotin 21 est le plus faible et le plus lent. Lorsqu'un Diablotin est occupé par l'avènement de ce qu'il surveille, les autres Diablotins dorment et ne se réveillent que lorsque le Diablotin de garde a terminé sa tâche.

Lorsqu'on prépare un jeu ou un projet employant les Diablotins, il est avantageux de suivre les règles suivantes:

1. Essayer de donner à un QUAND Diablotin un travail (listeinstruction) qu'il peut effectuer le plus vite possible.

2. La meilleure façon d'utiliser les QUAND Diablotins est de donner l'instruction FVIT Ø puis d'avoir recours à une procédure de support qui vérifie l'état de chaque Tortue.

3. La procédure de support ne doit pas faire autre chose que de surveiller l'occurrence d'une certaine condition et d'appeler une procédure de collision si cette dernière se produit.

4. La procédure de collision doit toujours protéger de l'occurrence d'une collision la Tortue surveillée par un Diablotin, sinon la Tortue peut être prise sur une ligne et éventuellement sortir de ses limites.

Exemples:

A chaque fois que le levier change de position (éventualité numéro 15), la procédure LEVIEROR est exécutée, ce qui permet de dessiner à l'aide du levier.

```
POUR LEVIEROR
SI (LEVIER Ø) < Ø [STOP]
FCAP 45 * LEVIER Ø
AV 5
LEVIEROR
FIN

QUAND 15 [LEVIEROR]
```

Le programme suivant met la Tortue en mouvement. Lorsque le bouton du levier est pressé (éventualité numéro 3), la Tortue rebondit comme un ressort.

```
POUR JOUE
VE MT BC
REPETE 4 [AV 1ØØ DR 9Ø]
LC FPOS [5Ø 5Ø]
QUAND 3 [BONDIR 1ØØ]
FVIT 1Ø
FIN

POUR BONDIR :VIT
SI :VIT < 1 [STOP]
AV :VIT ATTENDS 5Ø RE :VIT
BONDIR :VIT/2
FIN

JOUE
```

Voici une série de procédures qui dessinent un carré et y enferment quatre Tortues:

```
POUR PREPARE
DESIGNE [Ø 1 2 3] VE MT LC
FNC Ø FCC Ø 12Ø
DEMANDE Ø [FPOS [-5Ø -5Ø] BC REPETE 4->
 [AV 1ØØ DR 9Ø] LC]
DEMANDE Ø [FPOS [-2Ø -2Ø]]
DEMANDE 1 [FPOS [-2Ø 2Ø]]
DEMANDE 2 [FPOS [2Ø -2Ø]]
DEMANDE 3 [FPOS [2Ø 2Ø]]
CH [DR 9Ø * QUI]
FIN

POUR TACHED
QUAND Ø [FVIT Ø]
QUAND 4 [FVIT Ø]
QUAND 8 [FVIT Ø]
QUAND 12 [FVIT Ø]
FIN

POUR GUET
SI VIT = Ø [TROUVE]
GUET
FIN

POUR TROUVE
SI CONDP Ø [DEMANDE Ø [RE 1Ø DR 18Ø]]

SI CONDP 4 [DEMANDE 1 [RE 1Ø DR 18Ø]]

SI CONDP 8 [DEMANDE 2 [RE 1Ø DR 18Ø]]

SI CONDP 12 [DEMANDE 3 [RE 1Ø DR 18Ø]

FVIT 3Ø
FIN
```

```
PREPARE
TACHED
GUET
```

La procédure PREPARE met les quatre Tortues en place dans le carré. La procédure TACHED (tâche de diablotin) assigne un travail à chaque QUAND Diablotin. GUET est une procédure de support qui appelle la procédure de collision TROUVE.

REPETE commande

REPETE n listeinstruction

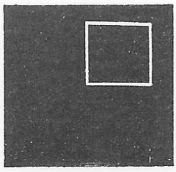
Cette commande recommence l'exécution d'une liste d'instructions le nombre de fois indiqué. Il y a erreur si n est négatif; s'il n'est pas un entier, Logo l'y tronquera.

Exemples:

```
REPETE 4 [AV 8Ø DR 9Ø]
```

dessine un carré ayant un côté de 8Ø pas de Tortue.

```
REPETE 4 [AV 8Ø DR 9Ø]
```



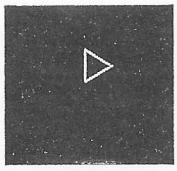
```
REPETE 5 [EC HASARD 2Ø]
```

écrit 5 nombres au hasard de Ø à 19.

La procédure suivante dessine des polygones:

```
POUR POLY :COTE :ANGLE
REPETE 36Ø / :ANGLE [AV :COTE DR :ANG->
LE]
FIN

POLY 5Ø 12Ø
```



```
POLY 5Ø 12Ø
```

RETOURNE, RT commande

RETOURNE objet

Cette commande ne s'utilise qu'à l'intérieur d'une procédure, jamais au niveau supérieur. Elle fait d'objet la sortie de la procédure et remet le contrôle d'exécution à la procédure appelante. RETOURNE est une commande mais la procédure qui la contient est une opération parce qu'elle est faite pour retourner quelque chose. Comparer avec STOP.

Exemples:

```
POUR MARK.TWAIN
RT [SAMUEL CLEMENS]
FIN

EC PH MARK.TWAIN [EST UN AUTEUR CELE->
BRE]
SAMUEL CLEMENS EST UN AUTEUR CELEBRE
```

La procédure suivante, ELEMENT, retourne le :Nième élément de la liste:

```
POUR ELEMENT :N :OBJ
SI VIDEP :OBJ [RT "]
SI :N = 1 [RT PREMIER :OBJ]
RT ELEMENT :N-1 SP :OBJ
FIN

RELIE "VOYELLES [A E I O U Y]
EC ELEMENT 2 :VOYELLES
E

EC ELEMENT 5 :VOYELLES
U

EC ELEMENT 7 :VOYELLES
```

La procédure suivante vérifie si la première entrée fait partie de la seconde et retourne VRAI ou FAUX selon le cas. C'est ainsi qu'on crée un prédicat.

```
POUR PARTIE :PART :TOUT
SI VIDEP :PART [RT "VRAI]
SI MEMBREP PREMIER :PART :TOUT [RT PA->
RTIE SP :PART :TOUT] [RT "FAUX]
FIN

EC PARTIE [W E] [A E I O U Y]
FAUX

SI PARTIE [I E] [A E I O U Y] [EC "V->
OYELLES]
```

VOYELLES

SI commande ou opération

SI pred listeinstruction
SI pred listeinstruction1 listeinstruction2

La première entrée, pred, est un prédicat ou une condition que SI vérifie comme étant VRAI ou FAUX. Si pred est VRAI, la listeinstruction1 s'effectue. Si pred est FAUX, la listeinstruction2 s'accomplit. Rien ne se produit s'il n'y a pas de listeinstruction2.

Dans l'un ou l'autre cas, si la listeinstruction choisie retourne quelque chose, SI donne la même chose. Quand la liste ne retourne rien, SI ne retourne rien. Lorsque cette primitive est utilisée avec une seule listeinstruction suivie, sur la même ligne, d'une autre commande, Logo livre un message d'erreur.

Exemples:

La procédure DECIDE est écrite de trois manières équivalentes. Les deux premières utilisent SI comme une commande. La première version avec deux entrées, la seconde avec trois. La dernière version de DECIDE utilise SI comme opération à trois entrées.

SI employée comme commande:

```
POUR DECIDE
SI Ø = HASARD 2 [RT "OUI]
RT "NON
FIN

POUR DECIDE
SI Ø = HASARD 2 [RT "OUI] [RT "NON]
FIN
```

SI employée comme opération:

```
POUR DECIDE
RT SI Ø = HASARD 2 ["OUI] ["NON]
FIN
```

N'importe laquelle des définitions précédentes donnera le même résultat, soit OUI, soit NON:

```
EC DECIDE
OUI
```

SI peut être employée à l'intérieur d'une autre condition utilisant SI. Par exemple:

```
POUR POSITIF :NOMBRE
SI NOMBREP :NOMBRE [SI :NOMBRE > Ø [E->
C [NOMBRE POSITIF]] [EC [NOMBRE NEGAT->
IF]]] [EC [PAS UN NOMBRE]]
FIN
```

STOP commande

STOP

La commande STOP arrête l'exécution de la procédure en cours et rend le contrôle à celle qui l'a appelée. Cette commande n'a de sens qu'à l'intérieur d'une procédure et non au niveau supérieur. Une procédure contenant STOP est une commande. Comparer à RETOURNE.

Exemple:

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

REBOURS 4
4
3
2
1
Ø
C'EST PARTI!
```

Chapitre 7

Les opérations logiques

Les prédicats sont des opérations ne retournant que VRAI ou FAUX. Pour la plupart, leur nom se termine par un P.

Il y a quelques prédicats Logo dont les entrées doivent être VRAI ou FAUX. Ce sont les opérations logiques. Leur nom ne se termine pas en P. Les concepteurs du Logo d'ATARI ont retenu les noms traditionnels ET, OU et NON pour ces opérations logiques. Elles servent à combiner les prédicats en expressions logiques, de la même façon que les opérations arithmétiques forment des expressions arithmétiques.

Aussi vrai que les opérations arithmétiques ne reçoivent et ne retournent que des valeurs numériques, les opérations logiques n'acceptent et ne retournent que VRAI ou FAUX.

Les entrées des opérations logiques sont habituellement des prédicats. Vous les retrouverez au fil des chapitres suivants.

Prédicats	Chapitres
CLEP	8
CONDP	6
EGALP	2
LEVIERB	8
LISTEP	2
MANETTEB	8
MEMBREP	2
MOTP	2
NOMBREP	2
NOMP	3
VIDEP	2
VISIBLEP	1
<	4
=	4
>	4

ET opération

ET pred1 pred2
(ET pred1 pred2 pred3...)

Peut recevoir deux entrées ou plus. ET retourne VRAI si toutes ses entrées sont VRAI; sinon, retourne FAUX.

Exemples:

```
EC ET "VRAI "VRAI
VRAI

EC ET "VRAI "FAUX
FAUX

EC ET "FAUX "FAUX
```

```
FAUX

EC (ET "VRAI "VRAI "FAUX "VRAI)
FAUX

EC ET 5 7
7 NI VRAI NI FAUX

EC ET (CC 1) = Ø FOND = Ø
FAUX
```

(L'opération infixe = retourne VRAI ou FAUX, conformément aux exigences de ET.)

La procédure suivante, DECIMALP, dit si l'entrée est un nombre décimal ou pas:

```
POUR DECIMALP :CHIFFRE
RT ET NOMBREP :CHIFFRE VERIFIE :CHIFF->
RE
FIN

POUR VERIFIE :CHIFFRE
SI VIDEP :CHIFFRE [RT "FAUX]
SI EGALP PREMIER :CHIFFRE ". [RT "VRA->
I]
RT VERIFIE SP :CHIFFRE
FIN

EC DECIMALP 17
FAUX

EC DECIMALP 17.Ø
FAUX
```

Remarquez que Logo considère comme un entier tout nombre se terminant par un point seul ou un point et un zéro.

```
EC DECIMALP 48.Ø98
VRAI

EC DECIMALP "STOP.
FAUX
```

FAUX mot spécial

```
FAUX
```

FAUX est une entrée spéciale pour ET, NON et OU.

NON opération

NON pred

Retourne VRAI si pred est FAUX; retourne FAUX si pred est VRAI.

Exemples:

```
EC NON EGALP "A "B
VRAI

EC NON EGALP "A "A
FAUX

EC NON "A = PREMIER "CHIEN
VRAI

EC NON "A
A NI VRAI NI FAUX
```

Si MOTP n'existait pas comme primitive, elle pourrait être définie comme ceci:

```
POUR MOT? :OBJ
RT NON LISTEP :OBJ
FIN
```

La procédure suivante dit si l'entrée est <<un mot qui n'est pas un nombre>>:

```
POUR VRAIMOT :OBJ
RT ET MOTP :OBJ NON NOMBREP :OBJ
FIN

EC VRAIMOT CAP
FAUX

EC VRAIMOT "CHAT
VRAI

EC VRAIMOT CRAYON
VRAI
```

OU opération

OU pred1 pred2
(OU pred1 pred2 pred3...)

OU retourne VRAI si au moins une de ses entrées est VRAI; sinon, retourne FAUX.

Exemples:

```
EC OU "VRAI "VRAI
```

```
VRAI

EC OU "VRAI "FAUX
VRAI

EC OU "FAUX "FAUX
FAUX

EC OU 5 7
7 NI VRAI NI FAUX
```

La procédure MONTAGNES dessine des <<montagnes>>:

```
POUR MONTAGNES
VE
DR 45
SOUSMONTAGNES
FIN

POUR SOUSMONTAGNES
AV 5 + HASARD 1Ø
SI OU YCOR > 5Ø YCOR < Ø [FCAP 18Ø - ->
CAP]
SOUSMONTAGNES
FIN
```

MONTAGNES

VRAI mot spécial

VRAI

VRAI est une entrée spéciale pour ET, NON et OU.

Chapitre 8

Le monde extérieur

Ce chapitre décrit les primitives qui permettent de communiquer avec divers appareils périphériques par l'entremise de votre ordinateur domestique ATARI. Ces appareils comprennent le clavier, l'écran et les appareils de commande tels les leviers. Si vous utilisez un téléviseur ou un moniteur muni d'un contrôle de volume, vous pouvez aussi tirer profit des primitives musicales du Logo d'ATARI, SON et FENV.

L'écran de l'ordinateur domestique ATARI contient 24 lignes de texte, à 38 caractères chacune. L'écran peut ne servir qu'au graphique ou, au contraire, ne servir qu'au texte. Vous pouvez aussi partager l'écran, utilisant les 19 lignes supérieures pour des fins graphiques; le texte apparaît sur les 5 dernières lignes. Lorsque vous démarrez Logo, l'écran est entièrement disponible au texte. Le curseur sur l'écran de texte est similaire à la Tortue de l'écran graphique. Vous pouvez produire des caractères partout sur l'écran en déplaçant d'abord le curseur à l'endroit voulu.

En plus des primitives décrites dans ce chapitre, les commandes SAUVE, RAMENE, FLIS et FEC sont aussi des primitives de communication avec le monde extérieur. Elle sont décrites au chapitre 1Ø.

CLEP opération

CLEP

Retourne VRAI s'il y a au moins un caractère attendant d'être lu sur le clavier ou sur tout autre périphérique déterminé par FLIS; sinon, retourne FAUX.

Exemple:

La procédure suivante fait avancer la Tortue à petits pas. Dès que vous pressez la touche D, la Tortue effectue un DR 1Ø. Lorsque vous pressez le G, la Tortue tourne suivant la commande GA 1Ø.

```
POUR CONDUIRE
AV 2
SI CLEP [TOURNE LISC]
CONDUIRE
FIN

POUR TOURNE :DIR
SI :DIR = "D [DR 1Ø]
SI :DIR = "G [GA 1Ø]
FIN
```

ECC commande

ECC objet

ECC (écris complet) écrit objet sur l'écran, suivi d'un retour de chariot. Si objet est une liste, elle est reproduite avec ses crochets. Comparez avec EC et TAPE.

Exemples:

```
ECC "A
A

ECC "A ECC [A B C]
A
[A B C]

TAPE "A TAPE [A B C]
AA B C

EC "A EC [A B C]
A
A B C
```

ECRANG commande

ECRANG

ECRANG (écran graphique) consacre tout l'écran au graphique. Seuls les dessins de la Tortue seront visibles; ce que vous taperez sera invisible. Cependant, Logo tiendra compte de vos instructions. Le texte reviendra à l'écran lorsque vous demanderez ECRANP ou ECRANT.

Si Logo désire vous faire part d'un message d'erreur et que l'écran est voué tout entier au graphique, il commutera l'état de celui-ci en ECRANP (écran partagé).

La combinaison CTRL F produit le même effet que ECRANG. De plus, CTRL F peut être ordonné pendant qu'une procédure est en cours, ce qui n'est pas le cas pour ECRANG. Pour utiliser la commande ECRANG, vous devez attendre l'apparition du symbole d'invite (?).

ECRANP commande

ECRANP

ECRANP (écran partagé) sépare l'écran pour laisser les 19 lignes supérieures à la Tortue et les 5 dernières au texte. L'état d'écran partagé surviendra dès que vous taperez une commande graphique.

La combinaison des clés CTRL S procure le même effet que

ECRANP. Voir aussi ECRANT et ECRANG.

ECRANT commande

ECRANT

ECRANT (écran texte) voue l'écran entier au texte. Le champ de la Tortue deviendra invisible jusqu'à ce qu'une commande graphique soit émise. La combinaison des clés CTRL T est équivalente à ECRANT. De plus, CTRL T opère même en cours de procédure, alors qu'il aurait fallu attendre le symbole d'invite pour taper ECRANT. Voir aussi ECRANG et ECRANP.

ECRIS, EC commande

ECRIS objet
(ECRIS objet1 objet2...)

Ecrit son ou ses entrées sur l'écran, suivies d'un RETOUR. Les crochets extérieurs des listes ne sont pas imprimés. Comparez avec TAPE et MONTRE.

Exemples:

```
ECRIS "A
A

ECRIS "A ECRIS [A B C]
A
A B C

(ECRIS "A [A B C])
A A B C

ECRIS [ ]


POUR REPRODUIRE :MESSAGE :FOIS
SI :FOIS < 1 [STOP]
EC :MESSAGE
REPRODUIRE :MESSAGE :FOIS - 1
FIN

REPRODUIRE [C'EST VENDREDI!] 4
C'EST VENDREDI!
C'EST VENDREDI!
C'EST VENDREDI!
C'EST VENDREDI!
```

FCURSEUR commande

FCURSEUR position

FCURSEUR (fixe curseur) sert à positionner le curseur. La position est une liste de deux nombres. Le premier élément est le numéro de la colonne, le second est le numéro de la ligne. Les lignes sont numérotées de Ø à 23 et les colonnes de Ø à 36.

Ce sera une erreur de proposer un nombre hors de ces limites, ou d'offrir, comme entrée, un nombre qui ne soit pas un entier. Notez que la colonne 38 est réservée à l'usage de -> (flèche de continuation de ligne).

Exemple:

```
FCURSEUR [35 12]
```

Le curseur se fixe au milieu de l'écran et à l'extrême droite.

FENV commande

FENV voix durée

FENV produit la forme de la représentation d'une onde sonore qui réduit le volume d'une voix déterminée (Ø ou 1) de 1 unité, et ce à chaque durée (nombre de soixantièmes de seconde). La durée, si non déterminée, est Ø, ce qui rend inopérante cette modification, et conséquemment tinte avec la sonorité caractéristique d'un ordinateur.

Exemple:

```
POUR TONØ :DUR
SON Ø 44Ø 15 :DUR
FIN

POUR TON1 :FR :DUR
SON 1 :FR 15 :DUR
FIN

POUR DECOMPTE
TONØ 12Ø TON1 11Ø 3Ø TON1 22Ø 3Ø
TONØ 6Ø TON1 33Ø 3Ø TON1 448 3Ø
FIN

FENV Ø 6
FENV 1 2
REPETE 6 [DECOMPTE]
```

LEVIER opération

LEVIER numérolevier

Retourne un nombre compris entre -1 et 7 représentant la

position du levier.

L'entrée doit être Ø, 1, 2 ou 3, soit le numéro du levier en opération. Toute autre entrée est une erreur.

Si le levier n'est pas connecté ou se trouve à sa position initiale (vous ne l'avez pas bougé), LEVIER retourne -1. Voir l'appendice C du manuel Introduction à la programmation via le graphique Tortue pour des exemples.

LEVIERB opération

LEVIERB numérolevier

Retourne VRAI si le bouton de ce levier est pressé; sinon, retourne FAUX. L'entrée doit être Ø, 1, 2 ou 3; il y a un maximum de 4 leviers. Toute autre entrée est une erreur. Si aucun levier n'est connecté, LEVIERB retourne FAUX. Voir l'appendice C du manuel Introduction à la programmation via le graphique Tortue pour des exemples.

LISC opération

LISC

LISC (lis caractère) retourne le premier caractère lu provenant d'un appareil périphérique ou du clavier. Ce caractère peut même être une touche CTRL. Si aucun caractère n'est prêt à être lu, LISC attendra que l'usager en tape un. Ce caractère n'est pas repris sur l'écran. Si la position finale d'un fichier a été atteinte pendant sa lecture, LISC retourne le mot vide. Voir aussi CLEP.

Exemple:

La procédure suivante permet à l'utilisateur d'exécuter certaines commandes à l'aide d'une touche: A exécute AV 5, D accomplit DR 1Ø et G, GA 1Ø.

```
POUR CONDUIRE
RELIE "CARAC LISC
SI :CARAC = "A [AV 5]
SI :CARAC = "D [DR 1Ø]
SI :CARAC = "G [GA 1Ø]
```

```
CONDUIRE
FIN
```

LISL opération

LISL

LISL (lis liste) retourne, sous forme de liste, la première ligne de mots lus par Logo et provenant du clavier ou d'un autre appareil périphérique. Si aucune liste n'est prête à être lue, LISL attendra que l'usager tape quelque chose au clavier. Si des listes ont déjà été introduites, LISL retourne la première série qui a été tapée mais pas encore lue. Tout ce que vous taperez sera repris sur l'écran. Si la position fin-de-fichier a été atteinte lors de sa lecture, LISL retourne la liste vide.

Exemples:

```
POUR L'USAGER
EC [QUEL EST VOTRE NOM?]
RELIE "LUI LISL
EC PH [BIENVENUE A LOGO,] :LUI
FIN

L'USAGER
QUEL EST VOTRE NOM
PIERRE
BIENVENUE A LOGO, PIERRE
```

MANETTE opération

MANETTE numéromanette

Retourne un nombre entre Ø et 227, représentant la rotation du cadran sur la manette spécifiée; numéromanette est un entier compris entre Ø et 7. Toute autre entrée est une erreur. Si aucune manette n'est branchée, MANETTE retourne Ø.

Exemple:

La procédure suivante vous permettra de dessiner en utilisant la manette Ø pour modifier le cap de la Tortue et la manette 1 pour la faire avancer.

```
POUR DESSINER
DR (MANETTE Ø) / 25.6
AV (MANETTE 1) / 25.6
DESSINER
FIN
```

MANETTEB opération

MANETTEB numéromanette

Retourne VRAI si le bouton de cette manette est pressé; sinon, retourne FAUX. Numéromanette est un entier compris entre Ø et 7 puisqu'il y a un maximum de 8 manettes. Toute autre entrée est une erreur. Si aucune manette n'est branchée, MANETTEB retourne FAUX.

Exemple:

La procédure DIRIGER permet de contrôler le mouvement de la Tortue par l'action du bouton. La Tortue décrira un cercle aussi longtemps que le bouton sera pressé et filera en ligne droite lorsque celui-ci sera relâché.

```
POUR DIRIGER
SI MANETTEB Ø [DR 5]
AV 2
DIRIGER
FIN
```

SON commande

SON voix fréquence volume durée

Génère une tonalité indiquée par voix (Ø ou 1), par l'entremise d'un haut-parleur. Fréquence est évaluée en hertz (cycles par seconde) et peut varier entre 14 Hz jusqu'au-dessus des fréquences audibles (44Ø Hz représente la note la). Le volume varie entre Ø et 15. La durée se mesure en multiples de 1/60^{e} de seconde et peut porter des valeurs entre Ø et 255.

Si un deuxième SON est attendu par la même voix, Logo attendra que le premier SON soit terminé.

Exemple:

```
POUR TINTER :FREQ
SON Ø :FREQ 15 15
EC :FREQ
TINTER :FREQ + 5Ø
FIN

TINTER 14
```

TAPE commande

TAPE objet
(TAPE objet1 objet2...)

Ecrit son ou ses entrées à l'écran, sans les faire suivre d'un retour de chariot. Les crochets extérieurs ne sont pas

reproduits. Comparez avec EC et ECC.

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 "\                    La barre inverse est suivie d'un espace.
FIN

POUR BOUGER
INVITE [COMBIEN DE PAS DEVRAIS\-JE FA->
IRE?]
AV PREMIER LISL
BOUGER
FIN

BOUGER
COMBIEN DE PAS DEVRAIS-JE FAIRE? 5Ø

COMBIEN DE PAS DEVRAIS-JE FAIRE? 37

COMBIEN DE PAS DEVRAIS-JE FAIRE? 2

COMBIEN DE PAS DEVRAIS-JE FAIRE? 1Ø8
```

VT commande

VT

VT (vide texte) vide l'écran de tout son texte et fixe le curseur dans le coin supérieur gauche de l'espace alloué au texte.

Chapitre 9

Gestion de l'espace de travail

L'espace de travail renferme les variables et les procédures que vous y avez introduites. Cela n'inclut pas les primitives.

Il y a plusieurs primitives qui vous permettent de voir ce qui se trouve dans votre espace de travail. Vous pouvez aussi, sélectivement, effacer des procédures s'y trouvant.

L'espace de travail est un site temporaire. Vos procédures et variables sont effacées lorsque vous éteignez votre ordinateur. Si vous voulez les conserver pour un usage ultérieur, vous devez les emmagasiner sur une disquette ou une cassette sous forme de fichiers. Voir le chapitre 10 pour plus d'informations sur les fichiers.

Notez que toutes les commandes débutant par EF vident le tampon d'édition. Si, après avoir donné une telle commande, vous tapez la commande EDITE sans entrée, l'éditeur ne contiendra aucune procédure.

EF commande

EF nom
EF listenoms

Efface la ou les procédures nommées de l'espace de travail. Cette commande n'affecte pas les procédures sauvées dans les fichiers.

Exemples:

EF "TRIANGLE
efface la procédure TRIANGLE.

EF [TRIANGLE CARRE]
efface les procédures TRIANGLE et CARRE.

EFN commande

EFN nom
EFN listenoms

EFN (efface nom) efface la ou les variables nommées de l'espace de travail.

Exemples:

EFN "LONGUEUR
efface la variable LONGUEUR.

EFN [LONGUEUR PI]
efface les variables LONGUEUR et PI.

EFNS commande

EFNS

EFNS (efface noms) efface toutes les variables de l'espace de travail.

EFPS commande

EFPS

EFPS (efface procédures) efface toutes les procédures de l'espace de travail.

EFTOUT commande

EFTOUT

EFTOUT (efface tout) efface toutes les procédures et variables de l'espace de travail. Cette commande libère aussi tous les noeuds du système. Assurez-vous que toutes les procédures et variables que vous désirez conserver sont sauvées dans des fichiers avant d'utiliser cette commande.

IM commande

IM nom
IM listenoms

IM (imprime) imprime la définition de la ou des procédures nommées. Vous ne pouvez pas imprimer la définition des primitives Logo.

Exemples:

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

IM [POLY SALUT]
```

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

POUR SALUT
EC [BONJOUR TOUT LE MONDE]
FIN
```

IMDEMON, IMD commande

IMDEMON numérocond

IMDEMON (imprime démon) imprime la condition et les instructions prévues pour le QUAND Diablotin numérocond; numérocond représente le numéro de la collision ou de l'éventualité (voir le tableau au début du chapitre 6). Voir QUAND pour la mise en place d'un QUAND Diablotin.

Exemples:

```
IMDEMON Ø
```

L'espace de travail ne contient pas de QUAND Diablotin Ø.

```
QUAND Ø [RE 1Ø]
IMD Ø
QUAND Ø [RE 1Ø]
```

IMDEMONS, IMDS commande

IMDEMONS

IMDEMONS (imprime démons) imprime les conditions et les instructions convenues pour tous les QUAND Diablotins.

Exemple:

```
IMDEMONS
QUAND Ø [RE 1Ø]
QUAND 3 [FVIT Ø]
```

IMNS commande

IMNS

IMNS (imprime noms) imprime le nom et la valeur de toutes les variables contenues dans l'espace de travail.

Exemple:

```
IMNS
RELIE "ANIMAL "CHAT
RELIE "LONGUEUR 3.98
RELIE "NOMS [LINDA MICHEL]
```

IMPS commande

IMPS

IMPS (imprime procédures) imprime la définition de toutes les procédures contenues dans l'espace de travail.

Exemple:

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

POUR SALUT
EC [BONJOUR TOUT LE MONDE]
FIN

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

IMTOUT commande

IMTOUT

IMTOUT (imprime tout) imprime la définition de toutes les procédures et la valeur de toutes les variables contenues dans l'espace de travail.

Exemple:

```
IMTOUT
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


RELIE "ANIMAL "CHAT
RELIE "LONGUEUR 3.98
RELIE "MONNOM "FRANCOIS
```

IMTS commande

IMTS

IMTS (imprime titres) imprime la ligne titre de toutes les procédures contenues dans l'espace de travail.

Exemple:

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

NOEUDS opération

NOEUDS

NOEUDS retourne le nombre de noeuds libres. Cela vous donne une idée de l'espace disponible dans l'espace de travail pour des procédures et des variables ou pour exécuter des procédures. Si vous désirez connaître le nombre exact de noeuds libres, exécutez la commande NOEUDS immédiatement après RECYCLE.

RECYCLE commande

RECYCLE

RECYCLE effectue un nettoyage, libérant ainsi le plus de noeuds possible. Si vous ne faites pas usage de la commande RECYCLE, cette collecte s'effectue automatiquement lorsque c'est nécessaire, mais prend à chaque fois au moins une seconde. Il est donc recommandé d'utiliser la commande RECYCLE avant d'exécuter une procédure où une interruption momentanée serait contre-indiquée. Voir NOEUDS.

Chapitre 10

Les fichiers

Les procédures et les variables présentes dans l'espace de travail sont effacées lorsque vous éteignez votre ordinateur. Si vous désirez les conserver pour un usage ultérieur, vous pouvez les emmagasiner sur une disquette ou une cassette. L'information est classée en fichiers. Vous gérez le contenu de chaque fichier.

Vous pouvez créer un fichier contenant une copie exacte de tous les caractères qui apparaissent à l'écran. Ce fichier <<journal>> créé par la commande FEC offre un compte rendu de l'interaction entre l'usager et l'ordinateur. Vous pouvez lire n'importe quel fichier, ligne par ligne, grâce à la commande FLIS.

Une imprimante est considérée comme un fichier d'un type spécial. Par exemple, vous pouvez sauvegarder les procédures et les variables de votre espace de travail en les envoyant à l'imprimante.

L'entrée d'une commande de fichier doit spécifier la nature de l'appareil périphérique concerné:

C: pour le lecteur de cassette;
D: pour le lecteur de disque;
Dn: pour le lecteur de disque n (n est le numéro du lecteur de disque, soit de 1 à 4);
P: pour papier de l'imprimante.

Lorsque D:, D1: ou Dn: (lecteur de disque) est utilisé, un nom de fichier doit suivre. Si vous utilisez un nom de fichier avec tout autre appareil périphérique, cette entrée sera ignorée. La seule exception consiste en la commande IMINDEX (imprime index) où l'entrée est utilisée seule.

Un nom de fichier peut contenir de 1 à 8 caractères avec une possibilité d'extension de 3 caractères. Le premier caractère d'un nom de fichier doit être une lettre. Toutes les lettres du nom du fichier, incluant l'extension, doivent être majuscules. Si vous faites usage de l'extension, un point doit séparer celle-ci du nom du fichier.

EFF commande

EFF périphérique:nomfichier

EFF efface le fichier appelé nomfichier sur la disquette. Si aucun fichier ne porte ce nom, Logo retourne un message d'erreur.

Le lecteur de disque est le seul appareil périphérique pouvant recevoir la commande EFF. Si vous avez plus d'un lecteur de disque, son numéro doit être spécifié.

Exemple:

```
EFF "D:BOITE
```
efface, sur la disquette, le fichier nommé BOITE.

FEC commande

```
FEC périphérique:nomfichier
FEC []
```

FEC (fixe écris) ouvre un fichier nommé périphérique:nomfichier et démarre le processus qui y envoie une copie de tous les caractères apparaissant sur l'écran de texte.

FEC []
FEC [] ferme le fichier.

Vous ne pouvez commander FEC qu'à un seul fichier à la fois mais vous pouvez utiliser FLIS et FEC en même temps. Si vous exécutez la commande FEC avec un appareil périphérique qui n'a pas été branché ou allumé après le démarrage de Logo, vous recevez un message d'erreur. Pour lire un fichier créé par FEC, utilisez la commande FLIS.

Exemples:

```
FEC "D:SEPT
```

ouvre un fichier appelé SEPT sur la disquette. Maintenant, tout ce qui apparaît sur l'écran de texte sera envoyé au fichier SEPT.

```
AV 4Ø
DR 9Ø
FEC "D:OCT
```

Le fichier SEPT est automatiquement fermé alors que s'ouvre le fichier OCT.

```
AV 3Ø
DR 45
FEC []
```

Le fichier OCT est fermé.

```
FLIS "D:SEPT
```

ouvre le fichier SEPT, prêt pour la lecture.

```
REPETE 3 [EC LISL]
AV 4Ø
DR 9Ø
FEC "D:OCT
```

Tout le contenu du fichier SEPT apparaît à l'écran.

```
FLIS "D:OCT
REPETE 3 [EC LISL]
AV 3Ø
DR 45
FEC []

FLIS []
```

Tout le contenu du fichier OCT apparaît à l'écran.

FLIS commande

FLIS périphérique:nomfichier
FLIS []

FLIS (fixe lis) ouvre le fichier nomfichier et démarre le processus qui permet d'envoyer le contenu du nomfichier à l'espace de travail par l'entremise du périphérique. Nomfichier peut être un fichier de procédures ou un fichier créé par FEC. Une fois la commande FLIS donnée, LISC et LISL lisent l'information transmise par ce périphérique:nomfichier.

FLIS []
FLIS [] ferme le fichier lu.

Vous ne pouvez commander FLIS qu'à un seul fichier à la fois mais vous pouvez ouvrir un fichier pour y lire (FLIS) et y écrire (FEC) en même temps.

Exemples:

```
FLIS "D:OURS
REPETE 4 [EC LISL]
POUR YEUX
OEILD
OEILG
FIN
```

Les premières lignes du fichier OURS apparaissent à l'écran.

FLIS []

Le fichier est fermé. Maintenant, LISC et LISL opèrent avec le clavier.

Voir FEC pour d'autres exemples.

IMINDEX commande

IMINDEX périphérique:

IMINDEX (imprime index) imprime à l'écran le nom de tous les fichiers contenus sur la disquette, si périphérique: est un lecteur de disque. Si périphérique: est un lecteur de cassette ("C:), toutes les procédures et les variables sont inscrites.

Exemples:

IMINDEX "D:
imprime le nom des fichiers contenus sur la disquette dans le lecteur en fonction.

IMINDEX "D2:
imprime le nom des fichiers contenus sur la disquette dans le lecteur numéro 2.

IMINDEX "C:
imprime la définition et le nom de toutes les procédures ainsi que les variables contenues sur le fichier de la cassette.

RAMENE commande

RAMENE périphérique:nomfichier

Ramène le contenu de nomfichier dans l'espace de travail, comme si vous l'aviez tapé au clavier. Si nomfichier n'existe pas ou si vous envoyez la commande RAMENE à une imprimante, un message d'erreur vous est retourné. La touche ARRET interrompt RAMENE.

Une fois le fichier ramené, vous pouvez en vérifier le contenu en utilisant les primitives IM, IMTOUT, etc. (Voir le chapitre 9, Gestion de l'espace de travail.) Pour plus d'informations concernant l'usage de SAUVE et RAMENE avec un lecteur de cassette, voir le chapitre 5 du manuel Introduction à la programmation via le graphique Tortue.

Exemples:

```
EFTOUT
```

L'espace de travail est maintenant vide.

```
RAMENE "D1:OURS
YEUX DEFINIE
JEU DEFINIE
TIR DEFINIE

RAMENE "C:
YEUX DEFINIE
JEU DEFINIE
TIR DEFINIE
```

SAUVE commande

SAUVE périphérique:nomfichier

SAUVE crée un fichier nommé nomfichier et y emmagasine toutes les procédures et variables contenues dans l'espace de travail.

Ne jamais utiliser la touche ARRET pendant qu'un fichier est en train d'être sauvé. Vous perdriez le contenu de votre espace de travail.

Une bonne habitude consiste à vérifier le contenu de l'espace de travail avant d'utiliser SAUVE afin d'effacer les procédures et variables inutiles. Voir IMTS, IM, IMTOUT et EF au chapitre 9.

Exemples:

SAUVE "D:MARIO.ØØ1 sauvegarde le contenu de l'espace de travail dans un fichier portant le nom MARIO.ØØ1 sur la disquette.

SAUVE "C: sauvegarde le contenu de l'espace de travail sur une cassette. (Voir le chapitre 5 du manuel Introduction à la programmation via le graphique Tortue pour plus de détails sur la façon de sauver des fichiers sur une cassette.)

SAUVE "P: imprime le contenu de l'espace de travail par l'entremise de l'imprimante.

Chapitre 11

Primitives spéciales

Il y a quelques primitives spéciales qui peuvent affecter le système Logo lui-même. Elles vous donnent accès à la mémoire de l'ordinateur et vous permettent d'en changer le contenu. Ces primitives sont en même temps dangereuses car un usage inattentif peut détruire le contenu de l'espace de travail . Le cas échéant, vous devez démarrer Logo de nouveau. Le nom de ces primitives commence avec un point, en guise d'avertissement. Vous devriez sauvegarder votre espace de travail avant d'expérimenter ces primitives.

.APPELLE commande

.APPELLE n

.APPELLE transfère le contrôle à la sous-routine en langage machine débutant à l'adresse n (décimal).

.DEPOSE commande

.DEPOSE n octet

.DEPOSE inscrit octet à l'adresse-machine n (décimal).

Exemples:

Les procédures suivantes changent la dimension de la Tortue:

```
POUR GROSSE
.DEPOSE 53256 1
FIN

POUR PETITE
.DEPOSE 53256 Ø
FIN

POUR PLUS.GROSSE
.DEPOSE 53256 3
FIN
```

.EXAMINE opération

.EXAMINE n

.EXAMINE retourne le contenu de l'adresse-machine n (décimal).

.FRATIO commande

.FRATIO n

.FRATIO fixe à n le rapport entre la longueur apparente d'un pas vertical de Tortue et un pas horizontal; n doit être compris entre -2 et 2. L'usage de cette commande vide l'écran.

.FRATIO .5 fait en sorte que les pas verticaux de la Tortue sont deux fois plus courts que ses pas horizontaux.

.FRATIO devient nécessaire lorsque, sur certains écrans, les carrés ont l'aspect de rectangles. Un ratio de .8 convient à la plupart des écrans.

.PRIMITIVES commande

.PRIMITIVES

.PRIMITIVES imprime la liste complète des primitives Logo.

Appendice A

Messages d'erreurs

ARRET!

La touche ARRET a été pressée, interrompant l'exécution en cours.

PROCEDURE DEJA DEFINIE

POUR a reçu, comme entrée, le nom d'une procédure déjà définie.

PRIMITIVE EST UNE PRIMITIVE

POUR ou EDITE a reçu une primitive comme entrée.

')' IMPREVUE

Une parenthèse fermante n'a pas de parenthèse ouvrante qui lui corresponde. Une parenthèse fermante a été trouvée là où Logo attendait une entrée.

PERIPHERIQUE:NOMFICHIER INTROUVABLE

Le nom du fichier donné comme entrée de RAMENE ou FLIS n'existe pas.

MANQUE D'ENTREES POUR PROCEDURE

Survient lorsqu'une primitive ou une procédure requiert un plus grand nombre d'entrées.

MEMOIRE REMPLIE

Votre espace de travail est presque entièrement rempli. Il est préférable d'effacer quelques procédures et variables qui ne sont plus utiles.

PROCEDURE N'A RIEN RETOURNE A PROCEDURE

Une primitive ou une procédure nécessitant une entrée non reçue se trouve sur la même ligne qu'une autre procédure ou primitive.

PRIMITIVE N'AIME PAS OBJET COMME ENTREE

PRIMITIVE a reçu une entrée incorrecte.

MOT N'EST PAS RELIEE

Une variable dont la valeur n'a pas été déterminée a été utilisée.

NE PEUX OUVRIR PERIPHERIQUE:NOMFICHIER

SAUVE, RAMENE, FEC ou FLIS a reçu une entrée incorrecte. Par exemple, le périphérique n'a pas été spécifié.

NE SAIS QUE FAIRE DE OBJET

Un objet Logo a été retourné sans être précédé d'une commande.

OBJET NI VRAI NI FAUX

SI, ET, OU ou NON a reçu une entrée qui n'était pas un prédicat (un prédicat retourne VRAI ou FAUX).

NOMBRE TROP GRAND

Le résultat d'une opération arithmétique est supérieur à 1E98 ($1Ø^{98}$) ou inférieur à 1E-98 ($1Ø^{-98}$).

PROCEDURE NON DEFINIE

Logo a essayé d'exécuter PROCEDURE mais n'a pas trouvé sa définition.

PAS A CE NIVEAU

La commande STOP ou RETOURNE a été utilisée en dehors d'une procédure.

TROP D'ELEMENTS ENTRE ( )

Les parenthèses sont mal situées dans une instruction Logo. Par exemple, elles englobent plus d'une expression Logo.

Appendice B

Touches spéciales

Un astérisque (*) indique que cette commande d'édition fonctionne tant à l'extérieur qu'à l'intérieur de l'éditeur.

*ARRET	Interrompt ce que Logo exécute. En mode d'édition, les changements portés au tampon d'édition sont ignorés.
*CTRL ->	Déplace le curseur d'une position vers la droite.
*CTRL <-	Déplace le curseur d'une position vers la gauche.
CTRL ↑	Remonte le curseur à la ligne précédente dans l'éditeur.
CTRL ↓	Descend le curseur à la ligne suivante dans l'éditeur.
*CTRL 1	Arrête le déroulement des pages de l'écran jusqu'à ce que la combinaison des touches CTRL et 1 soit pressée à nouveau.
*CTRL A	Ramène le curseur au début de la ligne courante.
*CTRL E	Renvoie le curseur à la fin de la ligne courante.
*CTRL EFF/ARR	Efface le caractère que recouvre le curseur.
CTRL F	Consacre l'écran complet aux graphiques Tortue.
CTRL INSERE	Dans l'éditeur, ouvre une nouvelle ligne à la position du curseur sans le déplacer.
CTRL S	Divise l'écran: la partie supérieure est vouée aux graphiques, la partie inférieure au texte.
CTRL T	Consacre l'écran complet au texte.
CTRL V	Amène à l'écran la page suivante dans l'éditeur.
*CTRL VIDE	Elimine le texte, de la position du curseur jusqu'à la fin de la ligne courante.

CTRL W	Déroule à l'écran la page précédente dans l'éditeur.
CTRL X	Reporte le curseur au début de l'éditeur.
*CTRL Y	Insère le contenu du tampon réserve.
CTRL Z	Envoie le curseur la fin de l'éditeur.
*EFF/ARR	Efface le caractère à gauche du curseur.
*HAUT EFF/ARR	Efface le texte de la position du curseur jusqu'à la fin de la ligne.
HAUT INSERE	Ouvre une nouvelle ligne à la position du curseur.
QUITTE	Met fin au mode d'édition et retourne le contrôle au niveau supérieur.
*RETOUR	Signale la fin d'une ligne et déplace le curseur au début de la ligne suivante.
\ (barre inverse)	Demande à Logo d'interpréter le caractère qui suit tel quel, plutôt que de lui conserver son sens véritable. Vous devez utiliser la barre inverse avant [,], (,), +, -, *, /, =, <, > et la barre inverse elle-même.

D'autres touches spéciales sont décrites à la section Démarrage.

Appendice C

Instruments utiles

Les procédures regroupées ici en ordre alphabétique peuvent vous être utiles pour construire vos propres procédures. Pour quelques-unes, un exemple d'utilisation figure dans ce manuel (voir l'index). D'autres procédures apparaissent ici pour la première fois.

ABS
retourne la valeur absolue de son entrée.

```
POUR ABS :NOMBRE
RT SI :NOMBRE < Ø [-:NOMBRE] [:NOMBRE->
]
FIN
```

CLASSER
prend une liste de mots et les retourne en ordre alphabétique.

SUPERORDRE
les organise sous forme de liste.

```
POUR CLASSER :NOUV :LISTE
SI VIDEP :NOUV [RT :LISTE]
RELIE "LISTE INSERER PREMIER :NOUV :L->
ISTE RT CLASSER SP :NOUV :LISTE
FIN

POUR INSERER :A :L
SI VIDEP :L [RT MP [ ] LISTE :A [ ]]
SI AVANT :A PREMIER SP :L [RT MP INSE->
RER :A PREMIER :L SP :L]
RT MD INSERER :A DERNIER :L SD :L
FIN

POUR AVANT :A :B
SI OU VIDEP :A VIDEP :B [RT VIDEP :A]->

SI NON EGALP PREMIER :A PREMIER :B [R->
T (ASCII :A) < (ASCII :B)]
RT AVANT SP :A SP :B
FIN

POUR SUPERORDRE :L
SI VIDEP :L [RT [ ]]
RT (PH SUPERORDRE PREMIER :L PREMIER ->
SP :L SUPERORDRE DERNIER :L)
FIN
```

Essayez ceci:

```
RELIE "LISTEMOTS CLASSER [A D E F T ->
C Z] [ ]

EC SUPERORDRE :LISTEMOTS
A C D E F T Z
```

Tapez ensuite:

```
RELIE "LISTEMOTS CLASSER [FIN BAL BO->
A] :LISTEMOTS
EC SUPERORDRE :LISTEMOTS
A BAL BOA C D E F FIN T Z
```

COPIEDEF
copie la définition d'une <<ancienne>> procédure sous un <<nouveau>> nom. COPIEDEF "CA "CARRE copie la définition de CARRE et donne à ce duplicat le nom de CA. Notez que COPIEDEF utilise les procédures DEFINIS et TEXTE

```
POUR COPIEDEF :NOUV :ANCIEN
RELIE "ANCIEN TEXTE :ANCIEN
DEFINIS :NOUV SP SP PREMIER :ANCIEN S->
P :ANCIEN
FIN
```

DEFINIS
fait d'une liste d'instructions la définition d'un nom donné en entrée.

```
POUR DEFINIS :NOM :ENTREE :LISTE
FEC "D:PROG
EC (PH "POUR :NOM :ENTREE)
ECRIRE :LISTE
EC "FIN
FEC [ ]
RAMENE "D:PROG
EFF "D:PROG
FIN

POUR ECRIRE :LISTE
SI VIDEP :LISTE [STOP]
EC PREMIER :LISTE
ECRIRE SP :LISTE
FIN

DEFINIS "CARRE ":COTE [[REPETE 4 [AV->
 :COTE DR 9Ø]]]
```

donne à CARRE cette définition:

```
POUR CARRE :COTE
REPETE 4 [AV :COTE DR 9Ø]
FIN
```

DETRUIS.DEM
efface tous les QUAND Diablotins si le nombre 21 est donné comme entrée. (Le tableau des collisions et des éventualités se trouve à la page 95)

```
POUR DETRUIS.DEM :DEMON
SI :DEMON < Ø [STOP]
QUAND :DEMON [ ]
DETRUIS.DEM :DEMON - 1
FIN
```

DIVISEURP
indique si la première entrée est un diviseur de la seconde.

```
POUR DIVISEURP :A :B
RT Ø = RESTE :B :A
FIN
```

ITEM
retourne le :Nième élément d'un mot ou d'une liste.

```
POUR ITEM :N :OBJET
SI VIDEP :OBJET [RT "]
SI :N = 1 [RT PREMIER :OBJET]
RT ITEM :N-1 SP :OBJET
FIN
```

LEQUEL
retourne, à l'inverse de ITEM, l'emplacement d'un élément dans une liste. LEQUEL "C [A B C] retourne 3.

```
POUR LEQUEL :MEMBRE :LISTE
SI VIDEP :LISTE [RT Ø]
SI :MEMBRE = PREMIER :LISTE [RT 1]
RT 1 + LEQUEL :MEMBRE SP :LISTE
FIN
```

POINT
place un point sur l'écran à la position [x y] donnée en entrée.
Notez que la procédure laisse la Tortue dans son état initial.

```
POUR POINT :POS
DEMANDE PREMIER QUI [POINT1 POS :POS ->
CRAYON VISIBLEP]
FIN

POUR POINT1 :AVANT :POS :CRAYON :VISIBLEP
CT LC
FPOS :POS
BC AV Ø LC
FPOS :AVANT
EXECUTE MP :CRAYON [ ]
SI :VISIBLEP [MT]
FIN
```

RENVOLTORTUE
vide l'écran de toutes ses Tortues, laissant seulement la Tortue Ø
dans sa forme régulière.

```
POUR RENVOLTORTUE
DESIGNE [Ø 1 2 3] VE
FFOR Ø CT
DESIGNE Ø MT
FIN
```

TANTQUE
répète une liste d'instructions tant que :CONDITION est VRAI.

```
POUR TANTQUE :CONDITION :LISTEINSTRUC->
TIONS
SI NON EXECUTE :CONDITION [STOP]
EXECUTE :LISTEINSTRUCTIONS
TANTQUE :CONDITION :LISTEINSTRUCTIONS->

FIN
```

TEXTE
retourne sous forme de liste la définition d'une procédure dont le
nom est donné en entrée. TEXTE "CARRE pourrait retourner [[POUR
CARRE :COTE] [REPETE 4 [AV :COTE DR 9Ø]]]

```
POUR TEXTE :NOM
FEC "D:PROG
IM :NOM
```

```
FEC [ ]
FLIS "D:PROG
RT LISLIGNE LISTE LISL "
FIN

POUR LISLIGNE :TX
RELIE "LIGNE LISL
SI [FIN] = :LIGNE [EFF "D:PROG RT :TX->
]
RT LISLIGNE MD :LIGNE :TX
FIN
```

TOUJOURS
exécute une liste d'instructions jusqu'à ce que la touche ARRET soit enfoncée ou qu'une coupure d'alimentation de l'ordinateur survienne.

```
POUR TOUJOURS :LISTEINSTRUCTIONS
EXECUTE :LISTEINSTRUCTIONS
TOUJOURS :LISTEINSTRUCTIONS
FIN
```

Appendice D

Espace mémoire

Les procédures et les variables Logo occupent de l'espace; l'exécution d'une procédure en exige davantage.

Peut-être voulez-vous savoir comment l'espace est utilisé et comment le conserver. D'une façon générale, il est inutile de vous inquiéter outre mesure d'économie d'espace. Composez plutôt vos procédures le plus clairement et le plus élégamment possible. Toutefois, nous reconnaissons que le Logo d'ATARI possède une mémoire limitée. Cet appendice décrit comment l'espace est alloué et la façon de l'économiser.

Fonctionnement

Dans Logo, l'espace est distribué en noeuds dont chacun a une longueur de 5 octets. Tous les objets et procédures Logo sont composés de noeuds. Logo en utilise aussi lorsqu'il travaille. L'interpréteur est au courant des noeuds libres prêts à l'usage. Lorsqu'il n'y a plus de noeuds libres, Logo met automatiquement en marche un récupérateur de mémoire qui recherche parmi tous les noeuds ceux qui ne sont plus utilisés.

Par exemple, au cours de l'exécution de l'expression suivante:

```
RELIE "NOMBRE 7
```

NOMBRE est assigné à deux noeuds qui contiennent la valeur 7.

Après l'exécution de

```
RELIE "NOMBRE 9Ø
```

les noeuds contenant la valeur 7 peuvent être réutilisés, et le seront effectivement à la prochaine exécution du récupérateur de mémoire. Ce récupérateur entre en fonction automatiquement lorsque c'est nécessaire mais la commande Logo RECYCLE permet son exécution à tous moments.

L'opération NOEUDS retourne le nombre de noeuds libres; toutefois, si vous désirez connaître la quantité réelle d'espace disponible, il est recommandé d'utiliser l'instruction suivante:

```
RECYCLE ECRIS NOEUDS
1259
```

Utilisation de l'espace

Chaque mot Logo utilisé n'est mis en mémoire qu'une fois: toutes les occurrences de ce mot sont en fait des pointeurs vers ce mot. Un mot utilise deux noeuds, plus un noeud pour chaque groupe de deux lettres dans son nom.

Un nombre, qu'il soit entier ou décimal, prend deux noeuds (exposant et mantisse). Une liste utilise un noeud pour chaque élément (plus la taille

de l'élément lui-même).

Suggestions pour économiser l'espace

L'utilisation d'abréviations peut vous sembler un bon moyen d'économiser de l'espace lorsque vous rédigez vos procédures, mais ceci les rend difficiles à lire. Employez plutôt les moyens suivants:

1. Réécrivez votre programme. Remplacez les sections répétitives du programme par des procédures.

2. Vous pouvez économiser de l'espace dans Logo en évitant de créer de nouveaux noms. Les noms des variables locales des procédures peuvent être les mêmes que celles d'autres procédures. Les noms de primitives et de procédures peuvent aussi servir de noms de variables.

3. Il faut noter que les erreurs d'orthographe, les erreurs de dactylographie et les mots qui ne sont plus utilisés ne sont pas détruits.

```
ECRUS "FOULE
ECRUS NON DEFINIE

EMBRASSER
EMBRASSER NON DEFINIE
```

Les mots FOULE, ECRUS et EMBRASSER sont créés et ne disparaîtront pas. Toutefois, si un mot n'a pas de valeur ni de définition de procédure, il ne sera pas transcrit dans un fichier. Ainsi, si l'espace vient à manquer et qu'il y a beaucoup de ces mots (parfois appelés atomes vraiment inutiles), vous pouvez envoyer votre espace de travail dans un fichier puis redémarrer Logo. Enfin, ramenez le contenu de votre espace de travail.

Appendice E

Interprétation

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 interprétation. Cet appendice vous permettra de comprendre l'interprétation des lignes. Pour constater l'effet de l'interprétation, tapez une ligne à l'intérieur d'une définition de procédure à l'aide de la commande POUR et voyez sa transcription par l'éditeur Logo.

Délimiteurs

Un mot est habituellement délimité par des espaces. Comprenons qu'il doit y avoir un espace avant et après le mot; il est ainsi séparé 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, pour découvrir la façon dont cette ligne est interprétée:

```
SI 1<2[ECRIS(3+4)/5][ECRIS :X+6]
```

tapez

```
POUR SAVOIR
SI 1<2[ECRIS(3+4)/5][ECRIS :X+6]
FIN

ED "SAVOIR
```

A l'écran, vous trouverez ceci:

```
POUR SAVOIR
SI 1 < 2 [ECRIS ( 3 + 4 ) / 5] [ECRIS->
 :X + 6]
FIN
```

Pour traiter l'un quelconque des caractères mentionnés ci-dessus comme un caractère alphabétique normal, il faut le faire précéder d'une barre inverse "\". Par exemple:

```
ECRIS "SAN\ FRANCISCO
SAN FRANCISCO
```

Procédures sous forme infixe

Les caractères =, <, >, +, -, *, / sont des procédures en forme infixe. Elles sont traitées comme des procédures à deux entrées, mais le nom de la procédure doit se trouver entre les deux entrées.

Crochets et parenthèses

Le crochet ouvrant "[" et le crochet fermant "]" indiquent le début et la fin d'une liste ou d'une sous-liste.

Les parenthèses ( ) regroupent les éléments à volonté plutôt que de laisser à Logo le soin de le faire à sa façon. Elles permettent aussi de varier le nombre d'entrées permises pour certaines primitives.

Si la fin d'une ligne Logo est atteinte (par l'usage de la touche RETOUR) et que des crochets ou des parenthèses n'ont pas été fermés, toutes les sous-listes ou expressions 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 rencontre un crochet fermant sans trouver le crochet ouvrant correspondant, il arrête l'exécution de la ligne ou de la procédure. Par exemple:

```
]ECRIS "ABC
```

Logo imprime une ligne vide.

Guillemets et délimiteurs

Habituellement, vous devez faire usage de la barre inverse devant les caractères [,], (,), +, *, -, /, =, <, > et la barre inverse elle-même. Mais le premier caractère après un guillemet (") ne requiert pas de barre inverse pour être considéré comme tel. Ainsi:

```
ECRIS "*
*
```

Si un délimiteur occupe une position autre que cette première place, il doit être précédé d'une barre inverse. Par exemple:

```
ECRIS "****
MANQUE D'ENTREES POUR *
```

Les crochets sont la seule exception à cette règle. Même après un guillemet, les crochets doivent être précédés d'une barre inverse.

```
ECRIS "[

NE SAIS QUE FAIRE DE [ ]

ECRIS "\[
[
```

Le signe moins

La façon dont le signe moins "-" est interprété n'est pas évidente. Le problème provient du fait que ce seul caractère cumule trois fonctions:

1. Lié à un nombre, il indique que celui-ci est négatif, comme dans -3.

2. Lié à une entrée, il est appelé le moins unaire et retourne l'inverse additif de l'entrée, comme dans -XCOR ou -:DISTANCE.

3. Dans une expression à deux entrées, il retourne leur différence, comme dans 7 - 3 ou XCOR - YCOR.

L'interpréteur tend à dénouer cette ambiguïté en se référant aux règles suivantes:

1. Si "-" précède immédiatement un nombre et suit un délimiteur (incluant un espace) sauf la parenthèse fermante ") ", le nombre est interprété comme une valeur négative. Cela explique les faits suivants:

   ECRIS 3 * -1 s'interprète comme 3 fois moins 1

   ECRIS 3*-4 s'interprète comme 3 fois moins 4

   PREMIER [- 3 4] retourne -

   PREMIER [-3 4] retourne -3

2. Si "-" est précédé d'une expression numérique, il agit comme "-" infixe.

   ECRIS 3-4 rend -1
   ECRIS XCOR - YCOR

3. Si "-" n'est pas précédé d'une expression numérique, il agit comme moins unaire.

   EC -XCOR
   EC -(3+4)

Appendice F

Code ASCII

code décimal	caractère	code décimal	caractère
Ø		3Ø	
1		31	
2		32	espace
3		33	
4		34	
5		35	
6		36	
7		37	
8		38	
9		39	
10		40	
11		41	
12		42	
13		43	
14		44	
15		45	
16		46	
17		47	
18		48	
19		49	
20		50	
21		51	
22		52	
23		53	
24		54	
25		55	
26		56	
27		57	
28		58	
29		59	

code décimal	caractère	code décimal	caractère
60	<	9Ø	Z
61	=	91	[
62	>	92	\
63	?	93	]
64	@	94	^
65	A	95	_
66	B	96	♦
67	C	97	a
68	D	98	b
69	E	99	c
7Ø	F	1ØØ	d
71	G	1Ø1	e
72	H	1Ø2	f
73	I	1Ø3	g
74	J	1Ø4	h
75	K	1Ø5	i
76	L	1Ø6	j
77	M	1Ø7	k
78	N	1Ø8	l
79	O	1Ø9	m
8Ø	P	11Ø	n
81	Q	111	o
82	R	112	p
83	S	113	q
84	T	114	r
85	U	115	s
86	V	116	t
87	W	117	u
88	X	118	v
89	Y	119	w

code décimal	caractère	code décimal	caractère
120	x	150	|
121	y	151	┬
122	z	152	┴
123	♣	153	▌
124	|	154	└
125	▒	155	RETOUR
126	◀	156	↑
127	▶	157	↓
128	♥	158	←
129	├	159	→
130	|	160	espace
131	┘	161	!
132	┤	162	"
133	┐	163	#
134	╱	164	$
135	╲	165	%
136	◢	166	&
137	■	167	'
138	◣	168	(
139	■	169	)
140	■	170	*
141	▬	171	+
142	▁	172	,
143	■	173	—
144	♣	174	.
145	┌	175	/
146	▬	176	0
147	┼	177	1
148	●	178	2
149	▬	179	3

code décimal	càractère	code décimal	caractère
18Ø	4	21Ø	R
181	5	211	S
182	6	212	T
183	7	213	U
184	8	214	V
185	9	215	W
186	:	216	X
187	;	217	Y
188	<	218	Z
189	=	219	[
19Ø	>	22Ø	\
191	?	221	]
192	@	222	^
193	A	223	—
194	B	224	♣
195	C	225	a
196	D	226	b
197	E	227	c
198	F	228	d
199	G	229	e
2ØØ	H	23Ø	f
2Ø1	I	231	g
2Ø2	J	232	h
2Ø3	K	233	i
2Ø4	L	234	j
2Ø5	M	235	k
2Ø6	N	236	l
2Ø7	O	237	m
2Ø8	P	238	n
2Ø9	Q	239	o

code décimal	caractère	code décimal	caractère
24Ø	p	248	x
241	q	249	y
242	r	25Ø	z
243	s	251	♣
244	t	252	|
245	u	253	↖
246	v	254	←
247	w	255	▶

CODE ASCII *

Les caractères blancs inclus dans les carrés noirs représentent l'affichage vidéo normal. Les caractères noirs dans les .carrés blancs sont en vidéo inverse.

* Une version étendue du code ASCII est utilisée pour contenir les caractères spéciaux de l'ordinateur ATARI.

Appendice G

Vocabulaire Logo

Graphiques Tortue
AVANCE, AV
BC
CAP
CC
CH
COULEUR
CRAYON
CT
DECRISFOR
DEFFOR
DEMANDE
DESIGNE
DROITE, DR
EDFOR
ENROULE
FCAP
FCC
FCT
FENETRE
FFOND
FFOR
FNC
FOND
FORME
FPOS
FVIT
FX
FY
GAUCHE, GA
GC
IC
LC
MT
NC
NETTOIE
ORIGINE
POS
QUI
RECULE, RE
VE
VISIBLEP
VIT
XCOR
YCOR

Mots et listes
ASCII
CAR
COMPTE
DERNIER
EGALP
LISTE
LISTEP
MD
MEMBREP
MOT
MOTP
MP
NOMBREP
PH
PREMIER
SD
SP
VIDEP
obj1 = obj2

Variables
CHOSE
NOMP
RELIE

Opérations arithmétiques
ARRONDIS
COS
ENT
HASARD
PRODUIT
RC
REHASARD
RESTE
SIN
SOMME
a + b
a - b
a * b
a / b
a < b
a = b
a > b

Définition et édition de procédures
EDITE, ED
EDNS
FIN
POUR

Instructions conditionnelles et contrôle d'exécution
ATTENDS
COLTC
COLTT
CONDP
EXECUTE
QUAND
QUAND [ ]
REPETE
RETOURNE, RT
SI
STOP

Opérations logiques
ET
FAUX
NON
OU
VRAI

Le monde extérieur
CLEP
ECRANG
ECRANP
ECRANT
ECRIS, EC
FCURSEUR
FENV
LEVIER
LEVIERB
LISC
LISL
MANETTE
MANETTEB
MONTRE
SON
TAPE
VT

Gestion de l'espace de travail
EF
EFN
EFNS
EFPS
EFTOUT
IM
IMDEMON, IMD

IMDEMONS, IMDS
IMNS
IMPS
IMTOUT
IMTS
NOEUDS
RECYCLE

Fichiers
EFF
FEC
FEC [ ]
FLIS
FLIS [ ]
IMINDEX
RAMENE
SAUVE

Primitives spéciales
.APPELLE
.DEPOSE
.EXAMINE
.FRATIO
.PRIMITIVES

Touches spéciales
ARRET
CTRL ->
CTRL <-
CTRL ↑
CTRL ↓
CTRL 1
CTRL A
CTRL E
CTRL EFF/ARR
CTRL F
CTRL INSERE
CTRL S
CTRL T
CTRL V
CTRL VIDE
CTRL W
CTRL X
CTRL Y
CTRL Z
EFF/ARR
HAUT EFF/ARR
HAUT INSERE
QUITTE
RETOUR
\ (barre inverse)

NOTE: Voir le glossaire pour les définitions et les entrées requises.

Appendice H

Glossaire

Note: Le symbole numéro (#) indique que cette procédure accepte un nombre indéfini d'entrées; si vous en donnez plus que la quantité indiquée, vous devez inclure l'expression toute entière entre parenthèses. Un astérisque (*) indique que cette commande d'édition fonctionne aussi bien à l'intérieur qu'à l'extérieur de l'éditeur. Pour connaître la définition des mots de données, voir page

.APPELLE n	Transfère le contrôle à une sous-routine en langage machine débutant à l'adresse n (décimal).
ARRONDIS n	Retourne n arrondi à l'entier le plus près.
ASCII car	Retourne le code ASCII de car.
ATTENDS n	Crée une pause de n 60ièmes de seconde.
AVANCE, AV distance	Déplace la Tortue de distance pas vers l'avant.
BC	Baisse le crayon de la Tortue.
CAP	Retourne le cap de la Tortue.
CAR n	Retourne le caractère dont le code ASCII est n.
CC numérocrayon	Retourne le nombre représentant la couleur du crayon numérocrayon.
CH liste	Fait en sorte que chaque Tortue exécute séparément les commandes contenues dans liste.
CHOSE nom	Retourne l'objet référé par nom.
CLEP	Retourne VRAI si une touche a été enfoncée mais pas encore lue.
COLTC numérotortue numérocrayon	Retourne le nombre symbolisant la collision entre numérotortue et numérocrayon.
COLTT numérotortue1 numérotortue2	Retourne le nombre symbolisant la collision entre numérotortue1 et numérotortue2.
COMPTE obj	Retourne le nombre d'éléments de objet.
CONDP numérocond	Retourne VRAI si la condition numérocond est effective.
COS n	Retourne le cosinus de n degrés.

COULEUR	Retourne le numéro représentant la couleur de la Tortue.
CRAYON	Retourne l'état du crayon (BC, LC, GC ou IC).
CT	Rend la Tortue invisible.
DECRISFOR numéroforme	Retourne une liste de 16 nombres; ces nombres correspondent aux bits de la forme.
DEFFORa numéroforme formespéc	Donne à numéroforme la forme de formespéc, la grille des bits.
DEMANDE numérotortue(s) liste	Demande à numérotortue(s) d'exécuter les instructions de liste.
.DEPOSE n octet	Ecrit octet à l'adresse n (décimal).
DERNIER obj	Retourne le dernier élément d'objet.
DESIGNE numérotortue(s)	Adresse les commandes suivantes à numérotortue(s).
DROITE, DR degrés	Tourne la Tortue de degrés degrés vers la droite (dans le sens des aiguilles d'une montre).
ECRANG (CTRL F)	Consacre tout l'écran au graphique.
ECRANP (CTRL S)	Divise l'écran: le haut pour le graphique, le bas pour le texte.
ECRANT (CTRL T)	Consacre tout l'écran au texte.
# ECRIS, EC obj	Ecrit obj suivi d'un retour de chariot (élimine les crochets extérieurs des listes).
EDFOR numéroforme	Démarre l'éditeur de forme Logo, montrant la forme numéroforme.
EDITE, ED nom(s)	Démarre l'éditeur Logo avec la ou les procédures nom(s).
EDNS	Démarre l'éditeur Logo avec toutes les variables de l'espace de travail.
EF nom(s)	Efface la ou les procédures nommées.
EFF périphérique:nomfichier	Efface nomfichier du périphérique.
EFN nom(s)	Efface la ou les variables de l'espace de travail.

EFNS	Efface toutes les variables de l'espace de travail.
EFPS	Efface toutes les procédures de l'espace de travail.
EFTOUT	Efface tout le contenu de l'espace de travail.
EGALP obj1 obj2	Retourne VRAI si ses entrées sont égales.
ENROULE	Enroule le champ de la Tortue autour des bords de l'écran. Vide l'écran.
ENT n	Retourne la partie entière de n.
# ET prédl préd2	Retourne VRAI si toutes ses entrées sont VRAI.
.EXAMINE n	Retourne le contenu de l'adresse n (décimal).
EXECUTE liste	Exécute liste; retourne ce que liste retourne.
FAUX	Entrée spéciale pour ET, NON, OU et SI.
FCC numérocrayon numérocouleur	Donne au crayon numérocrayon (Ø, 1 ou 2) la couleur numérocouleur.
FCAP degrés	Fixe le cap de la Tortue à degrés degrés.
FCT numérocouleur	Fixe le numérocouleur de la Tortue.
FCURSEUR pos	Amène le curseur à pos.
FEC périphérique:nomfichier	Démarre le processus qui achemine une copie de tous les caractères apparaissant à l'écran vers périphérique:nomfichier.
FEC [ ]	Ferme le fichier ouvert par FEC.
FENETRE	Transforme l'écran graphique en fenêtre sur le champ étendu de la Tortue. Vide l'écran.
FENV voix durée	Fixe la forme de la représentation de voix pour SON de façon à réduire le volume d'une unité à chaque durée.
FFOND numérocouleur	Fixe la couleur du fond à numérocouleur.
FFOR numéroforme	Fixe la forme de la Tortue à numéroforme.

FIN	Met fin à la définition de procédure débutée avec POUR.
FLIS périphérique:nomfichier	Fixe le périphérique:nomfichier duquel les sorties de LISC et LISL seront lues.
FLIS [ ]	Ferme le fichier ouvert par FLIS.
FNC numérocrayon	Fixe le crayon au numérocrayon (Ø, 1 ou 2).
FOND	Retourne le nombre représentant la couleur du fond.
FORME	Retourne le nombre représentant la forme de la Tortue actuelle.
FPOS position	Amène la Tortue à pos.
.FRATIO n	Fixe le rapport pas vertical / pas horizontal à n.
FVIT vitesse	Fixe la vitesse de la Tortue.
FX x	Déplace la Tortue horizontalement jusqu'à l'abscisse x.
FY y	Déplace la Tortue verticalement jusqu'à l'ordonnée y.
GC	Transforme le crayon en gomme à effacer.
GAUCHE, GA degrés	Tourne la Tortue vers la gauche de degrés degrés (dans le sens inverse aux aiguilles d'une montre).
HASARD n	Retourne un entier pris au hasard entre Ø et n - 1.
IC	Dépose le crayon inverseur.
IM nom(s)	Ecrit la définition de la ou des procédures nommées.
IMDEMON, IMD numérocond	Ecrit les numérocond des QUAND Diablotins actuellement en fonction.
IMDEMONS, IMDS	Ecrit tous les QUAND Diablotins actifs.
IMINDEX périphérique:	Affiche les noms de tous les fichiers contenus sur la disquette. Sur une cassette, affiche la définition des procédures et les noms contenus dans le fichier.

IMNS	Ecrit le nom et la valeur de toutes les variables.
IMPS	Ecrit la définition de toutes les procédures.
IMTOUT	Ecrit la définition de toutes les procédures et noms (variables).
IMTS	Ecrit la ligne titre de toutes les procédures.
LC	Lève le crayon de la Tortue.
LEVIER numérolevier	Retourne la position actuelle du numérolevier.
LEVIERB numérolevier	Retourne VRAI si le bouton de numérolevier est enfoncé.
LISC	Retourne le caractère lu par le périphérique en fonction (le clavier par défaut). Attend si nécessaire.
LISL	Retourne la ligne lue par le périphérique en fonction (le clavier par défaut). Attend si nécessaire.
LISTE obj1 obj2	Retourne une liste contenant ses entrées.
LISTEP obj	Retourne VRAI si obj est une liste.
MANETTE numéromanette	Retourne la rotation du cadran de numéromanette.
MANETTEB numéromanette	Retourne VRAI si le bouton est pressé sur numéromanette.
MD obj liste	Retourne une liste formée en ajoutant obj à la fin de liste.
MEMBREP obj liste	Retourne VRAI si obj fait partie de liste.
MONTRE obj	Ecris obj suivi d'un retour de chariot et incluant les crochets des listes.
# MOT mot1 mot2	Retourne un mot composé de ses entrées.
MOTP obj	Retourne VRAI si obj est un mot.
MP obj liste	Retourne une liste formée en ajoutant obj au début de liste.
MT	Rend la ou les Tortues visibles.

NC	Retourne le numéro du crayon utilisé (Ø, 1 ou 2).
NETTOIE	Efface l'écran graphique sans affecter l'état de la Tortue.
NOEUDS	Retourne le nombre de noeuds libres.
NOMBREP obj	Retourne VRAI si obj est un nombre.
NOMP nom	Retourne VRAI si nom a une valeur.
NON préd	Retourne VRAI si pred est FAUX.
ORIGINE	Amène la Tortue à [Ø Ø] et fixe le cap à Ø.
# OU préd1 préd2	Retourne VRAI si au moins une de ses entrées est VRAI.
# PH obj1 obj2	Retourne la liste de ses entrées.
POS	Retourne la liste des coordonnées de la position de la Tortue.
POUR nom (entrées)	Démarre la définition de la procédure nom.
PREMIER obj	Retourne le premier élément d'obj.
.PRIMITIVES	Ecrit la liste des primitives Logo.
# PRODUIT a b	Retourne le produit de ses entrées.
QUAND numérocond liste	Prépare un QUAND Diablotin et lorsque la condition numérocond est respectée, liste est exécutée.
QUAND numérocond [ ]	Congédie le ou les QUAND Diablotins en action.
QUI	Retourne le numéro de la ou des Tortues en action.
RAMENE périphérique:nomfichier	Ramène le fichier nommé nomfichier du périphérique jusqu'à l'ordinateur.
RC n	Retourne la racine carrée de n.
RECULE, RE distance	Déplace la Tortue de distance pas vers l'arrière.
RECYCLE	Exécute une récupération de la mémoire disponible.

REHASARD	Reproduit l'effet de HASARD.
RELIE nom obj	Fait référer nom à obj.
REPETE n liste	Exécute liste n fois.
RESTE a b	Retourne le reste de a divisé par b.
RETOURNE, RT obj	Retourne le contrôle à la procédure appelante, avec obj comme sortie.
SAUVE périphérique:nomfichier	Sauve le contenu de l'espace de travail sur périphérique.
SD obj	Retourne tout sauf le dernier élément de obj.
SI préd liste1 (liste2)	Si pred est VRAI, exécute liste1; sinon, exécute liste2 (s'il y a lieu).
SIN degrés	Retourne le sinus de degrés degrés.
SOMME a b	Retourne la somme de ses entrées.
SON voix fréq volume durée	Produit un son sur voix d'une fréquence fréq au volume pendant durée.
SP obj	Retourne tout sauf le premier élément d'obj.
STOP	Met fin à la procédure et retourne le contrôle à la procédure appelante.
# TAPE obj	Ecrit obj, laissant le curseur à la fin de la ligne imprimée.
VE	Efface de l'écran tout tracé de la Tortue et place la ou les Tortues à la position [Ø Ø], pointant vers le nord, cap Ø.
VIDEP obj	Retourne VRAI si obj est vide.
VISIBLEP	Retourne VRAI si la Tortue est visible.
VIT	Retourne la vitesse de la Tortue en fonction.
VRAI	Entrée spéciale pour ET, NON, OU et SI.
VT	Vide l'écran de texte.
XCOR	Retourne la valeur de l'abscisse de la Tortue.

YCOR	Retourne la valeur de l'ordonnée de la Tortue.
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 plus petit que b.
a > b	Retourne VRAI si a est plus grand que b.
obj1 = obj2	Retourne VRAI si obj1 est égal à obj2.

Touches spéciales

Touche ATARI (⅄) Touche vidéo inverse (◪)	Après l'usage de cette touche, tous les caractères apparaissent à l'écran en vidéo inverse.
* ARRET	Interrompt la procédure en cours. En mode d'édition, sort de l'éditeur en ignorant les modifications apportées.
* CTRL ->	Avance le curseur d'un espace vers la droite.
* CTRL <-	Recule le curseur d'un espace vers la gauche.
CTRL ↑	Remonte le curseur à la ligne précédente.
CTRL ↓	Descend le curseur à la ligne suivante.
* CTRL 1	Arrête le déroulement des pages de l'écran jusqu'à ce que la combinaison des touches CTRL et 1 soit pressée à nouveau.
* CTRL A	Ramène le curseur au début de la ligne courante.
* CTRL E	Renvoie le curseur à la fin de la ligne courante.
CTRL F	Consacre l'écran complet au graphique.
* CTRL EFF/ARR	Efface le caractère que recouvre le curseur.

CTRL INSERE	Ouvre une nouvelle ligne à la position du curseur.
CTRL S	Divise l'écran: le haut pour le graphique et le bas pour le texte.
CTRL T	Consacre l'écran complet au texte.
CTRL V	Amène la page suivante à l'écran dans l'éditeur.
* CTRL VIDE	Efface le texte à partir de la position du curseur jusqu'à la fin de la ligne courante.
CTRL W	Amène la page précédente à l'écran dans l'éditeur.
CTRL X	Reporte le curseur au début de l'éditeur.
* CTRL Y	Dans l'éditeur, insère le texte conservé dans le tampon réserve. A l'extérieur de l'éditeur, CTRL Y insère la dernière ligne Logo tapée.
CTRL Z	Envoie le curseur à la fin de l'éditeur.
* EFF/ARR	Efface le caractère à gauche du curseur.
F1, F2, F3, F4	Touches de contrôle du curseur qui peuvent être programmées.
* HAUT EFF/ARR	Elimine le texte, de la position du curseur jusqu'à la fin de la ligne.
HAUT INSERE	Ouvre une nouvelle ligne là où le curseur se trouve.
QUITTE	Complète l'édition et rend le contrôle au niveau supérieur.
REDEM	Redémarre Logo, effaçant le contenu de l'espace mémoire.
* RETOUR	Ouvre une nouvelle ligne à la position actuelle du curseur et renvoie ce dernier au début de cette nouvelle ligne.
\ (barre inverse)	Avise Logo d'interpréter le caractère qui suit littéralement, plutôt que de lui conserver son sens véritable. Vous devez utiliser la barre inverse devant [,], (,), +, -, *, =, >, <, et la barre inverse elle-même.

ATARI LOGO

MANUEL DE REFERENCE

Matériel nécessaire :
un Ordinateur ATARI de 16 Ko MEV
option : un lecteur de cassette ATARI
pour utilisation avec la cassette
et le livre HATIER
« 35 Activités Logo pour Atari »

Le LOGO est un langage évolué, conçu par Seymour Papert, spécialiste de l'intelligence artificielle appliquée aux techniques d'éducation à l'Université du M.I.T. à Boston.

Adapté en français et présenté sous forme de cartouche, le LOGO d'ATARI® est un outil qui facilite la communication entre les utilisateurs et l'ordinateur. Il se prête aussi bien à l'initiation des enfants qu'à la réalisation d'applications très sophistiquées.

Dès le plus jeune âge, les enfants se passionnent pour les quatre petites tortues. Ils les font défiler, tourner en rond, changer de couleur, d'aspect, de direction. Ils les utilisent pour écrire ou faire des calculs savants. En tapant des instructions simples et compréhensibles en français, ils s'initient facilement à la programmation tout en développant des notions de calcul, de géométrie, de logique et de dessin.

Pour l'utilisateur confirmé, le langage LOGO permet de développer des programmes aussi divers que des jeux, de la création artistique ou des logiciels éducatifs.

Pour découvrir rapidement et facilement les énormes possibilités du langage LOGO, les Editions HATIER proposent un ouvrage où 35 Activités LOGO sont décrites avec, pour chacune d'entre elles, le programme LOGO correspondant expliqué et commenté. Une cassette comprenant 27 programmes accompagne le livre. Ces 27 programmes sont destinés à être utilisés dans les logiciels que vous réaliserez vous-même en langage LOGO.

Le LOGO d'ATARI®, de vrais amours de petites tortues dont l'éventail de possibilités est épatant.

La cartouche ATARI® LOGO est livrée avec son guide de référence et deux manuels :
INTRODUCTION A LA PROGRAMMATION VIA LE GRAPHIQUE TORTUE et le MANUEL DE REFERENCE ATARI® LOGO.

ATARI

P.E.C.F. ATARI 9/11, rue Georges Enesco 94008 - CRETEIL CEDEX
Ce produit est développé pour Atari Inc. par Logo Computer Systems Inc.

IMPRIME EN FRANCE