AMOS

LE CREATEUR

Manuel de l'Utilisateur

MANDARIN SOFTWARE

AMOS

Le Créateur

... par François Lionet

Mandarin/Jawx 1991

MANDARIN SOFTWARE

Conception et programmation	François Lionet
Coordinateur du Projet	Richard Vanner
Auteur du Manuel	Stephen Hill
Editeur technique	Peter Lee

Composition du Manuel et Visual Eyes

Traduction Salford Translations Ltd

Si vous avez un ennui quelconque avec ce logiciel, veuillez écrire à: Service Clientèle, Mandarin Software, Europa House, Adlington, Macclesfield, Cheshire, SK10 4NP, Angleterre.

Distributeurs français:
UBI SOFT
8-10 rue de Valmy
93100 Montreuil-sous-Bois
France

ISBN #1-872984-60-5 Première édition juin 1990 Edition révisée en février 1991

Présentation de la traduction d'AMOS

Bienvenue à la version traduite d'AMOS le Créateur. Nous avons pensé qu'il est indispensable d'aider les utilisateurs français; dès que la version anglaise fut sortie, nous avons entrepris un gros travail de traduction.

Certains termes ont été laissés en anglais sur les disquettes mais tous les fichiers importants que vous utiliserez souvent ont été traduits et le manuel vous donne au besoin les équivalents en français.

Nous avons travaillé dur pour que le manuel soit exempt d'erreur mais s'il vous arrive d'en trouver ou si vous pouvez suggérer des améliorations, écrivez à:

UBI SOFT
8-10 rue de Valmy
93100 Montreuil-sous-Bois
France

Comment a-t-il été réalisé?

AMOS Basic a été conçu et programmé par François Lionet. Ses idées ingénieuses et ses travaux inspirants ont permis le développement d'un langage de programmation de haut niveau et, à notre opinion, il est de loin le meilleur qu'il soit sur l'Amiga à l'heure actuelle.

Les programmes suivants ont été utilisés lors du développement d'AMOS:

Devpac II Assembler - Hisoft
Deluxe Paint III - Electronic Arts
Pix Mate - Progressive Peripherals & Software
Cross-DOS - Consultron
Mini Office Professional Communications - Database Software

Mandarin Software tient à exprimer ses remerciements à toutes les personnes suivantes qui ont apporté leurs collaboration tout au long du développement d'AMOS.

Alistair Brimble, Aaron et Adam Fothergill de Shadow Software, Peter Hickman, Rico Holmes, Commodore UK pour ses configurations internationales du clavier (et l'Amiga), Commodore France pour avoir participé à la résolution des problèmes rencontrés avec l'A1000, l'A3000 et le CDTV, 17-Bit Software pour les échantillons et les démos, Martyn Brown pour avoir participé à la création d'une super bibliothèque du Domaine Publique, Virus Free PD pour le Soundtracker, Simon Cook pour ses commentaires constructifs et la localisation des bogues, Lee Alex et tous les autres développeurs d'AMOS pour leur aimable collaboration et tout ceux qui ont attendu patiemment la sortie de ce logiciel. Nous espérons que vous êtes d'avis que cela valait bien la peine d'attendre.

Ce manuel a été dactylographié sur Apple Macintosh en utilisant WriteNow et mis en page à l'aide de Quark.

Copyright

AMOS vous permet de créer des logiciels très impressionnants. Il est très important que vous mentionniez AMOS dans vos programmes dans les termes suivants par exemple: “Rédigé par Jacques Durand en utilisant AMOS” et si possible intégrez-y le sprite AMOS.

Tables des matières

1 Présentation

Bienvenue au monde passionnant de AMOS, *Le Créateur!* Comme vous savez, l'Amiga est un ordinateur vraiment sensationnel. Toute cette puissance est pour la première fois entre vos mains.

En Septembre 1988, Mandarin Software a sorti le STOS Basic pour le ST. Cette date fut un évènement historique: il fut le premier langage de programmation à s'être classé au top niveau des graphismes ludiques sur ST. STOS a été entièrement réécrit pour donner naissance à AMOS Basic pour l'Amiga. AMOS Basic offre un champs de plus de 500 commandes; la puissance de la plupart d'entre elles est stupéfiante. Vous pouvez faire bouger un écran ou animer un sprite en utilisant une seule instruction Basic.

AMOS n'est pas simplement une autre version du Basic; c'est un système de création ludique spécialisé qui est écrit dans son propre langage d'animation (AMAL). Les programmes AMAL sont exécutés 50 fois par seconde en utilisant un puissant système d'interruption. Ils peuvent être utilisés pour produire n'importe quel effet, depuis les séries d'attaque d'un jeu d'arcade jusqu'aux scrollings super fluides. Votre programme Basic peut en même temps être en train d'exécuter une tâche tout à fait différente!

Quelque soient vos connaissances en programmation, AMOS a quelque chose à vous offrir. Si vous n'avez jamais écrit de jeu, la perspective d'en créer un tout premier peut être intimidante. Mais n'oubliez pas que les grands classiques sont pour la plupart des programmes simples dotés d'une ou deux particularités originales; il vous suffit de regarder Tetris par exemple. La force de votre jeu va plutôt dépendre de la qualité de vos idées que de vos compétences en programmation. Grâce à AMOS, vous pouvez désormais créer des jeux qui semblent professionnels en un tout petit effort. Tout ce qu'il vous faut vraiment est de l'imagination.

Si vous avez écrit un jeu en AMOS Basic, ne le gardez pas pour vous. Mandarin Software tient à publier tout programme qui a été écrit en AMOS. Ne vous inquiétez pas si votre programme n'est pas très au point. Seule la valeur de vos idées importe et peut vous apporter la clé du succès en vous offrant la possibilité de devenir auteur professionel de jeux. N'hésitez pas à nous envoyer vos programmes. Mandarin serait également ravi de lire vos commentaires et vos suggestions concernant le système AMOS. Plusieurs fonctions d'AMOS ont été développées à partir d'idées qui nous ont été envoyées par des utilisateurs de STOS. Adressez votre courrier à l'attention de Richard Vanner, Development Manager, Mandarin Software, Adlington Park, Adlington, Macclesfield, SK10 4NP, Angleterre.

Et enfin quelques mots de l'homme qui a écrit ce logiciel, François Lionet, un Français exceptionnellement doué. François a commencé sa carrière en tant que vétérinaire et a crée le STOS Basic d'origine dans ses loisirs.

Dedicace

J'aimerais dédicacer ce programme à ma femme Carine, pour sa patience les soirs de stress!

Preface

Bienvenue dans le monde de l'AMOS! Vous avez entre les mains de résultat de 14 mois de programmation intense, dans lequel je peux vous l'assurer, j'ai mis le meilleur de moi même!

L'Amiga est une fantastique machine, tant au niveau matériel qu'au niveau logiciel. Le programme système est très puissant, trop même: il reste extrèmement complexe et difficile à apréhender. Un débutant ne pourra jamais exploiter à fond les capacités de la machine.

En programmant l'AMOS, j'ai voulu oser ce que personne n'avait fait jusque là dans un langage: attaquer directement les circuits en outrepassant (dans une certaine mesure bien sûr) le programme système. Cette approche permet à l'AMOS d'offrir la rapidité du langage machine dans un langage simple et structuré: le Basic. Les effets de scrolling, de barres de couleurs, de bobs, sprites, double buffer et autres dual playfield sont maintenant très simples à mettre en oeuvre... Les résultats obtenus en quelques heures feront palir de jalousie les programmeurs ayant passé des journées entières à écrire leur démo!

L'intelligence du système de l'Amiga permet d'obtenir de tels effets tout en restant totalement conforme aux règles de programmation de la machine. AMOS est parfaitement multitache, et peut être utilisé en même temps que d'autres programmes.

Avant, pour programmer un jeu, il fallait passer plus des trois quart du temps à fabriquer les routines de base, assurant l'affichage d'une image, le dessin des personnages etc... Avant l'AMOS, faire un jeu impliquait beaucoup de programmation stérile.

Bonne nouvelle: ce travail pénible est fait une fois pour toute! Plus besoin de se casser la tête sur les listes copper, les registres du blitter et les librairies systèmes! Vous pouvez maintenant vous concentrer sur les parties réellement importantes de votre jeu: le scénario, les graphismes et la musique...

L'AMOS est extensible. Il peut suivre l'évolution de votre machine préférée. Vous pouvez même rajouter vos propres instructions au 600 déjà disponibles, grâce au système d'extensions. Les routines musicales par exemple, peuvent facilement être modifiées par vos soins, si vous programmez en assembleur. Le listing de ces routines se trouve sur la disquette EXTRAS. Vous pouvez l'étudier et le modifier à loisir.

Un dernier mot sur -devinez quoi- le piratage. Le couplet habituel. On ne dira jamais assez que copier une disquette c'est participer à l'extinction des vrais ordinateurs au profit des consoles de jeu: les éditeurs préfèrent investir dans une cartouche qui ne peut pas être dupliquée. Alors si vous voulez encore trouver un ordinateur comprenant un clavier et un lecteur de disquette dans quelques années, ne copiez plus!

D'autre part, être un utilisateur légal et enregistré auprès de Mandarin vous permet de bénéficier de nombreux avantages: support, information sur les nouveaux produits, clubs, domaine public AMOS, etc...

Voila c'est fini, merci de m'avoir lu. J'espère que vous passerez de nombreuses heures créatives avec votre Amiga et l'AMOS!

François

2 Comment démarrer

AMOS Basic est un progiciel vraiment remarquable, capable de créer des jeux qui dépassaient auparavant vos rêves les plus fous. Toutes les puissantes fonctions qui rendent l'Amiga si attrayant ont été intégrées à ce fantastique système. Avec l'aide d'AMOS Basic, vous pouvez élaborer des programmes mettant à l'épreuve les capacités du programmeur le plus expert en matière de langage assembleur.
Par exemple, vous pouvez animer simultanément 56 sprites machine - sans effort! C'est un véritable exploit lorsque vous considérez que le hardware de l'Amiga ne vous en offre que huit.

S'il vous faut plus d'actions à l'écran, vous pouvez également utiliser le blitter. Les objets graphiques à manipuler peuvent être dessinés sur n'importe quel mode graphique, y compris le HAM! La seule limitation restreignant le nombre d'objets à l'écran est la capacité de la mémoire.

On peut afficher en même temps sur l'écran n'importe quelle combinaison de modes de résolution. Le défilement interne n'est plus seulement faisable, il devient un jeu d'enfant! Il existe une commande intégrée SCREEN OFFSET vous permettant d'exécuter directement toute l'opération.

En fait, la seule difficulté que présente le langage AMOS Basic est de savoir par où commencer! AMOS supporte plus de 500 commandes Basic, et si vous n'avez jamais utilisé le Basic, il se peut que vous vous sentiez un peu impressionné par la grandeur même de ce système. Quand vous êtes en territoire inconnu, il est toujours utile d'avoir un guide qui vous pilote et vous signale les principaux points de repère. C'est le but de ce chapitre.

Sauvegardez votre AMOS dès maintenant!

Avant de vous aventurer plus loin, il est essentiel que vous sauvegardiez tout le progiciel AMOS Basic sur des disquettes vierges. Cela protègera la disquette AMOS de tout risque d'erreur accidentelle. Et maintenant vous pouvez vous amuser avec le système autant que vous voulez sans courir le risque de détruire quelque chose d'important.

Même en envisageant le pire, nous serons heureux de remplacer votre disquette moyennant une somme administrative nominale. Mais vous serez évidemment privé de l'AMOS le temps qu'il soit de nouveau dupliqué.

La procédure d'installation varie selon la précision de votre équipement, mais elle est normalement l'affaire de quelques minutes.

Comment installer AMOS sur un système à un lecteur de disquettes

1 Préparez deux disquettes vierges qui conserveront vos copies en AMOS Basic.
2 Prenez votre disquette programme AMOS; en poussant l'onglet de protection en écriture vers la droite, vous verrez une petite ouverture. Cette onglet protège votre

disquette des erreurs malencontreuses qui peuvent se glisser lors du processus de duplication.

3 Démarrez votre Amiga avec votre disquette Workbench normale.
4 Introduisez votre disquette programme AMOS dans le lecteur interne et sélectionnez son icône à l'aide de la souris.
5 Choisissez maintenant l'option duplication de disquette à partir du menu Workbench et suivez les invites au fur et à mesure qu'elles apparaissent à l'écran. Si vous rencontrez un problème dans le déroulement de cette opération, consultez le manuel d'utilisation de l'Amiga qui a été livré avec votre ordinateur. Il vous donne une explication détaillée de toute la procédure.
6 Répétez les instructions 4 et 5 pour la disquette de données AMOS.
7 Finalement, placez vos copies AMOS Basic d'origine dans un endroit sûr.

Comment installer AMOS sur un système à deux lecteurs de disquettes

1 Préparez deux disquettes vierges qui conserveront vos copies AMOS Basic.
2 Prenez votre disquette programme AMOS; en poussant l'onglet de protection en écriture vers la droite, vous verrez une petite ouverture. Cette onglet protège votre disquette des erreurs malencontreuses qui peuvent se glisser lors du processus de duplication.
3 Démarrez votre Amiga avec votre disquette Workbench normale.
4 Introduisez votre disquette programme AMOS dans le lecteur interne et sélectionnez son icône à l'aide de la souris.
5 Introduisez une disquette vierge dans le deuxième lecteur.
6 Déplacez la souris de façon à ce que la disquette AMOS d'origine se positionne sur votre nouvelle disquette.
7 Il vous est maintenant donné l'option de copier AMOS sur la nouvelle disquette. Suivez les invites au fur et à mesure qu'elles apparaissent à l'écran pour effectuer la sauvegarde de la disquette.
8 Répétez les instructions 4 et 5 pour la disquette de données AMOS.
9 Finalement, placez vos copies AMOS Basic d'origine dans un endroit sûr.

Comment installer AMOS sur un disque dur

Ceci implique un programme d'installation spécial sur la disquette programme AMOS. La procédure est la suivante:

1 Initialisez votre Amiga depuis votre disque dur comme vous en avez l'habitude.
2 Ouvrez une fenêtre CLI ou SHELL.
3 Introduisez votre disquette programme AMOS dans le lecteur interne.
4 Positionnez le répertoire sur le lecteur interne en tapant:

CD Df0:

5 Vous pouvez maintenant taper l'instruction suivante après l'apparition de l'invite CLI:

6 AMOS se charge en mémoire et le programme d'installation du disque dur vous est présenté. Suivez simplement les invites pour installer AMOS directement sur votre disque dur.

Comment charger AMOS Basic

Comme vous pouviez vous y attendre, AMOS Basic peut être exécuté de bien différentes façons. Par exemple, vous pouvez charger AMOS directement depuis le Workbench en sélectionnant son icône avec le bouton gauche de la souris. Une fois que vous avez lancé AMOS de cette façon, vous pouvez revenir au Workbench en appuyant sur les touches Amiga et A du clavier.

Toutefois, en pratique, le Workbench consomme une mémoire précieuse qu'il serait plus judicieux de garder pour vos programmes Basic. Si vous êtes un utilisateur sérieux, vous allez sans doute préférer booter l'AMOS. Ceci vous permettra d'obtenir les meilleurs résultats possibles de votre système AMOS.

Pour charger l'AMOS Basic:

- Eteignez votre Amiga et attendez environ dix secondes.
- Introduisez une copie de sauvegarde de la disquette Programme AMOS (disquette 1) dans le lecteur interne.
- Mettez en marche votre Amiga. AMOS se chargera automatiquement dans la mémoire.
- Appuyez sur une touche pour éffacer la page de présentation et lancez le système AMOS.

Guide d'initiation AMOS

La première chose que vous voyez lorsque vous lancez AMOS Basic est la fenêtre de l'éditeur. Elle est extrêmement simple à utiliser, et si vous êtes un peu familier avec les ordinateurs, cela devrait vous sembler évident. N'hésitez pas à expérimenter autant que vous le souhaitez. L'éditeur AMOS est très intelligent et il est improbable que vous commettiez de graves erreurs.
Maintenant que vous avez vu la fenêtre de l'éditeur, il est temps d'explorer quelques-unes des fonctions qui font de l'AMOS un Basic différent.

Comment charger un programme

Nous allons commencer par vous montrer comment vous pouvez charger un des superjeux depuis la disquette de données AMOS. Nous allons prendre le jeu Number Leap par exemple:

- Introduisez la disquette de données AMOS dans le lecteur Df0: (le lecteur interne).
- Maintenez une touche Amiga enfoncée et appuyez sur la lettre L. Ceci amènera un sélecteur standard de fichier à l'écran.
- Cliquez sur le "bouton" lecteur de disquettes DF0 pour informer AMOS que vous avez

changé de disquette.
- Au centre du sélecteur de fichiers, vous trouverez une liste de programmes pouvant être chargés en AMOS Basic.
- Pour sélectionner le programme Number Leap, il suffit de positionner la flèche de la souris sur le fichier:

Number_Leap.AMOS

Le fichier que vous avez choisi sera ainsi mis en surbrillance.

- Une fois que vous avez choisi votre fichier, vous pouvez le charger dans la mémoire de l'Amiga en cliquant deux fois le bouton gauche de la souris. Votre jeu est maintenant sélectionné depuis la disquette de données AMOS et la page d'origine de l'éditeur réapparaît. Le contenu de cette fenêtre sera actualisé pour afficher le nouveau programme.
- Vous pouvez lancer ce programme en sélectionnant le "bouton" RUN se trouvant sur la zone principale du menu (ou en frappant la touche F1 de votre clavier si vous êtes paresseux).

La page de l'éditeur va maintenant disparaître complètement et Number Leap s'exécutera sous vos yeux. Une fois rassasie, vous pouvez sortir du jeu en appuyant simultanément sur les touches Ctrl et C.

Ctrl+C vous permet efficacement de sortir de la plupart des programmes AMOS. Elle peut être inhibée par le programme en utilisant la commande BREAK OFF, ce qui vous donne davantage de sécurité. Une fois sorti du programme, vous pouvez toujours revenir rapidement à l'éditeur en appuyant sur la barre d'espacement de votre clavier.

Comment éffacer un programme

Vous n'avez plus besoin maintenant du programme Number Leap et vous pouvez donc l'effacer de la mémoire en utilisant la commande NEW. Vous ne trouverez pas cette option au menu principal: elle a été placée dans un menu SYSTEME indépendant. Elle peut être amenée à l'écran en déplaçant la flèche de la souris sur la fenêtre du menu et en maintenant le bouton droit de la souris abaissé.

Pour effacer un programme:

- Assurez-vous que la flèche de la souris se trouve sur la zone du menu.
- Maintenez le bouton droit de la souris abaissé pour amener le menu SYSTEME à l'écran.
- Tout en appuyant sur le bouton, déplacez la flèche sur l'option NEW et sélectionnez-la à l'aide de la touche gauche de la souris. Vous pouvez également exécuter directement cette option depuis le clavier en appuyant sur Shift+F9.
- Tapez O pour valider l'opération ou tapez N pour suspendre son déroulement. (Vous pouvez également mettre ces options en surbrillance en utilisant la souris.)
- Si le programme que vous êtes en train d'utiliser n'a pas été sauvegardé, il vous sera demandé si vous voulez le conserver sur disquette.
- Si vous sélectionnez l'option Oui, un sélecteur de dossier AMOS s'affichera. Sinon,

votre programme sera complètement effacé.

Mode direct

Nous allons jeter un coup d'oeil rapide au mode direct. Ce mode constitue le coeur de l'AMOS Basic; il vous permet de mettre en pratique vos routines et d'en observer immédiatement les effets. Il est important de savoir que toutes les images, les sprites et la musique définis dans votre programme sont indépendants de la fenêtre éditeur.

Aussi, peu importe ce que vous faites en mode direct, vous pouvez retourner à votre listing en appuyant simplement sur une seule touche.

- Passez en mode direct en appuyant sur Escape. La fenêtre éditeur va disparaître et le principal écran-programme se présentera à vous.

Vers le bas de l'écran se trouve une petite fenêtre qui peut être utilisée pour passer aux commandes en mode direct. Tapez la ligne suivante et appuyez sur la touche Retour pour la valider:

```
Print "Votre nom"
```

Inscrivez votre nom entre les guillemets pour l'imprimer sur l'écran de l'Amiga.

Appuyez maintenant sur les flèches verticales de votre clavier pour déplacer la fenêtre dans la zone de l'écran. Comme vous pouvez le constater, la fenêtre en mode direct est totalement indépendante de l'écran-programme principal.

Animation!

Voilà tout ce qu'il y a à dire sur le mode direct. Mettons en pratique quelques-unes des instructions de sprites d'AMOS Basic. Avant d'utiliser ces commandes, il nous faut charger un jeu d'images de sprites en mémoire. Restez en mode direct et entrez les lignes d'instruction en alinéa et en caractère gras au fur et à mesure que vous les rencontrez.

Lister les fichiers de sprites à l'écran

Nous allons commencer par lister tous les fichiers disponibles de sprites à l'écran d'Amiga.

- Assurez-vous que la disquette de données AMOS est toujours dans le lecteur interne.
- Faites apparaître le répertoire du fichier de la disquette avec l'instruction:

```
Dir "AMOS_DATA:Sprites/"
```

Cette instruction fera apparaître les fichiers de sprites se trouvant sur la disquette de données AMOS. Ces fichiers contiennent toutes les images utilisant l'accessoire de création de sprites de la disquette programme AMOS.

L'éditeur de sprites comprend une foule de fonctions graphiques puissantes vous permettant très facilement de créer des séquences d'animation professionnelles dans vos jeux.

Charger un fichier de sprites

Nous pouvons maintenant charger ces sprites en utilisant la commande load. Les sprites seront chargés dans une banque de mémoire spéciale, par conséquent ne vous attendez pas à voir déjà des sprites apparaître! Chargeons les sprites utilisés dans le jeu Number Leap à l'aide de la commande suivante:

```
Load "AMOS_DATA:Sprites/Frog_Sprites.abk"
```

Si vous commettez une erreur, frappez la touche F1 pour revenir à la ligne précédente. Celle-ci peut alors être éditée en utilisant les touches du curseur et ré-exécutée en appuyant sur la touche Retour.

Chargeons également un fichier musique en utilisant une commande load similaire:

```
Load "AMOS_DATA:Music/Funkey.abk"
```

Afin de vérifier si les sprites et la musique ont été bien chargées dans la mémoire, nous allons rappeler l'instruction LISTBANK en tapant:

```
Listbank
```

Elle imprime à l'écran une ligne comme suit:

```
1 - Sprites          S:$0682B0 L:$000040
3 - Musique          S:$043878 L:$0081FE
```

Ne vous inquiétez pas si les chiffres ne correspondent pas à ceux que vous obtenez parce qu'ils varient selon la mémoire disponible. Le nombre de sprites que nous venons de charger peut vous être renvoyé directement par la fonction LENGTH.

```
Print Length (1)

64
```

Nous vous donnerons des lignes à entrer tout au long de ce manuel. Elles seront mis en surbrillance en caractères gras. Tout message venant de l'ordinateur sera affiché sous les lignes du programme en texte convivial.

Déterminer les couleurs des sprites

Chaque jeu d'images de sprites a sa propre palette de couleurs dont les valeurs sont conservées sur disquette. Puisque celles-ci peuvent être vraiment différentes des couleurs actuelles de l'écran, il est utile de pouvoir saisir (GRAB) les couleurs de la banque de sprite et de les copier sur un écran existant. Ceci est permis par la commande GET SPRITE PALETTE. Entrez la ligne suivante:

```
Get Sprite Palette
```

Toutes les couleurs de l'écran-programme principal changent immédiatement, mais la fenêtre en mode direct est aucunement affectée parce que le système AMOS lui a assigné une liste indépendante de codes couleur qui lui est propre.

Afficher un sprite

Les sprites peuvent être affichés n'importe où à l'écran en utilisant une commande AMOS Basic simple pour les sprites. Voici un exemple:

```
Sprite 8,129,50,62
```

Animer un sprite

Animons cet objet en utilisant le **L**angage d'**A**nimation **AM**OS (AMAL). AMAL est un système d'animation unique qui peut être utilisé pour bouger ou animer vos objets à une vitesse incroyable.

Veuillez noter que lorsque vous lancez les programmes de démonstration suivants, il est important de taper chaque instruction exactement comme elle apparaît, sinon vous pouvez faire une erreur de syntaxe inattendue.

```
Sprite 8,129,150,62
Amal 8,"Anim 0,(62,5)(63,5)(64,5);" : Amal On
```

Le programme ci-dessus anime un petit canard à l'écran. Tant qu'il est manipulé, le sprite peut être déplacé à l'aide de la commande SPRITE. Par exemple:

```
Sprite 8,300,50,
```

Déplacer un sprite

Et maintenant un peu de mouvement!

```
Sprite 8,129,150,62 : A$="Anim 0,(62,5)(63,5)(64,5);"
A$=A$+"Loop: Move 320,0,100; Move -320,0,100; Jump Loop"
Amal 8,A$ : Amal On
```

Ce programme anime le canard et le fait avancer ou reculer, en utilisant simplement ces trois lignes d'instruction!

Bien que les instructions entre les guillemets puissent ressembler au langage Basic, elles sont en fait écrites en AMAL. Tous les programmes en AMAL sont exécutés

50 fois par seconde et ils peuvent être exploités indépendamment de vos programmes en Basic pour produire des effets d'animation super fluides.

Juste pour vous prouver que votre AMAL est vraiment fantastique, frappez la touche Esc pour revenir à l'éditeur Basic. Retournez au mode direct au bout de quelques instants. Votre sprite sera toujours en train de traverser l'écran en sautant comme si rien ne s'était passé!

Musique maestro!

Le final, en avant la musique! Assurez-vous que vous êtes toujours en mode direct, montez le volume de votre moniteur et mettez la musique en marche à l'aide de la commande MUSIC en tapant:

```
Music 1
```

A propos, vous pouvez arrêter la musique à l'aide de la commande:

```
Music off
```

Le voyage continue

Avec un peu de chance, vous devez à présent avoir une assez bonne idée de la capacité de votre AMOS Basic. Jusqu'à maintenant, nous n'avons vu qu'une toute petite partie de la puissance de AMOS Basic. Tout au long du parcours d'initiation, vous découvrirez avec votre progiciel AMOS tout un monde nouveau riche de possibilités passionnantes.

AMOS Basic ne peut pas, bien évidemment, vous transformer en un expert-programmeur de jeux du jour au lendemain. Comme avec tout langage de programmation, il va vous falloir un peu de temps pour vous familiariser à tout le répertoire de commandes. C'est pourquoi, pour vous guider sur la route de votre apprentissage, nous terminerons ce chapitre par quelques conseils pratiques.

Conseils et suggestions

- La meilleure façon d'étudier votre AMOS est de créer des petits programmes pour animer les sprites, faire défiler les écrans, produire des tableaux de marquage des points. Une fois que vous avez pris un peu de confiance en vous-même, vous pourrez intégrer ces routines dans un vrai programme.
- Ne soyez pas intimidé par le volume même du langage AMOS Basic. En pratique, vous pouvez obtenir des effets prodigieux grâce à seulement quelques-unes des 500 commandes de votre AMOS. Commencez par maîtriser deux ou trois instructions telles que SPRITE et BOB, et avancez lentement au travers des chapitres. A mesure que vous progresserez, vous allez acquérir petit à petit une connaissance détaillée de votre système AMOS.
- Bien que nous ayions essayé de rendre ce logiciel aussi abordable que possible, nous estimons que la possession de solides bases en principes généraux de programmation en Basic est d'une valeur inestimable. Si vous n'avez aucune connaissance préalable du Basic, il serait préférable que vous achetiez un manuel qui

servira d'introduction tel que Apprendre à Programmer en Basic - **Claude Delannoy (Editions Eyrolles) ou Initiation au Basic - H. Lilen (Editions Radio).**

- Elaborez soigneusement vos jeux par écrit. Vous serez surpris par le nombre de problèmes que vous pourrez ainsi éviter dès les premières étapes de la conception. N'essayez jamais d'entreprendre de trop grands projets sans les avoir préparés à l'avance. Sans préparation, vous risquez à coup sûr de vous perdre à jamais!
- Lorsque vous écrivez un jeu, n'essayez pas de vous concentrer sur la qualité du jeu mais attachez plutôt de l'importance aux effets spéciaux. Le graphisme et la musique peuvent toujours être ajoutés plus tard si les idées s'avèrent être assez bonnes.
- Remplissez le coupon d'inscription et devenez immédiatement membre du club des utilisateurs d'AMOS! Nous vous enverrons régulièrement un bulletin d'informations vous apportant une source indispensable d'idées, de nouvelles et de conseils sur le système AMOS Basic. Vous pourrez également avoir accès à une logithèque du domaine publique en plein développement, comprenant entre autres des programmes sur les sprites, des échantillons et de la musique, tous pouvant être aisément intégrés aux programmes de votre création.

Mais AMOS ne s'arrête pas là et va bientôt vous surprendre avec des innovations fantastiques comme les extensions, les programmes utilitaires et même des livres. Si vous voulez jouer un véritable rôle dans l'amélioration de ce progiciel, adhérez au club AMOS **dès maintenant!** Nous serons enchantés de vous acceuillir.

3 L'éditeur

L'éditeur AMOS vous offre une gamme très étendue de fonctions d'édition. Non seulement il est exceptionnellement puissant, mais aussi s'en servir est un plaisir. Vous pouvez exécuter toutes les commandes soit directement depuis l'écran soit par l'intermédiaire d'un choix impressionnant de variantes simples sélectionnées au clavier. Il est en fait si convivial, que même si vous n'êtes que très peu familier avec les ordinateurs, vous pourrez sans doute l'utiliser directement après l'avoir sorti de sa boîte d'emballage.

Une des caractéristiques les plus remarquables de ce système est que la totalité du listing apparaît complètement indépendant de votre écran-programme principal. Ainsi, vous pouvez passer en un clein d'oeil de votre programme à la fenêtre de l'éditeur en appuyant sur une seule touche (ESCape).

Si vous disposez d'une grande mémoire, vous pouvez également charger simultanément plusieurs programmes AMOS Basic. Chaque programme peut être édité tout à fait indépendamment et il est possible de passer facilement d'un programme en mémoire à l'autre en appuyant tout simplement sur deux touches de l'éditeur.

La première chose que vous voyez après qu'AMOS ait été chargé en mémoire est un écran d'accueil classique. Bravo et encore bravo! Appuyez sur une touche pour faire disparaître cette fenêtre et lancer l'éditeur.

La fenêtre du menu

En haut de l'écran, vous verrez une fenêtre menu contenant la liste des commandes disponibles. Elle représente la porte d'entrée à toute les fonctions puissantes d'édition de l'AMOS Basic. Vous pouvez exécuter rapidement toutes les commandes en déplaçant la flèche de la souris sur un objet puis en frappant le bouton gauche de la souris. Chaque commande est également affectée à une touche de fonction particulière.

Il existe également un certain nombre d'autres menus complétant le menu principal. Le menu le plus important de tous est le menu SYSTEME. Il peut être visualisé à l'écran, soit en maintenant le bouton droit de la souris abaissé, soit en appuyant sur la touche majuscule de votre clavier.

Le menu SYSTEME comprend un choix d'importantes commandes du système telles que LOAD, SAVE, NEW, etc. Comme avec le menu principal, vous pouvez exécuter toutes les options soit en utilisant le bouton gauche de la souris, soit en appuyant sur la touche de fonction appropriée. Voici quelques exemples:

F1	Démarrer le programme en cours
F2	Tester votre programme pour localiser les erreurs de syntaxe
Shift+F1	Charger un programme (Egalement Amiga+L)
Shift+F3	Sauvegarder un programme (Amiga+S)

Juste en dessous de la fenêtre du menu se trouve une seule ligne d'information renfermant une gamme de renseignements utiles.

La ligne d'information

I **L=1** **C=1** **Text=**40000 **Chip=**91000 **Faste=**0 **Edit:**exemple

Les repères se trouvant sur l'extrême gauche de l'écran représentent le mode de l'éditeur (**I**nsertion ou **S**uperposition). Il y a également une indication de la ligne (**L**igne) et de la colonne (**C**olonne) que vous êtes en train d'éditer. Un nombre se trouve aux côtés de chacun des repères suivants.

TEXT: Mesure la quantité de mémoire qui a été allouée à la fenêtre de l'éditeur. Vous pouvez l'ajuster lorsque vous vous trouvez dans AMOS Basic en utilisant une simple commande SET BUFFER que vous sélectionnez au MENU RECHERCHE.

CHIP: Détient la quantité de mémoire vidéo, accessible aux circuits de l'Amiga. Ne paniquez pas si vous n'avez qu'un Amiga 500 doté d'une mémoire non étendue et donc un peu avide de mémoire. Il existe plusieurs façons d'augmenter considérablement cette valeur dans vos programmes Basic (voir le paragraphe sur CONSERVER LA MEMOIRE pour plus de précisions).

FAST: Liste la quantité de mémoire RAPIDE qui a été installée dans votre ordinateur. AMOS préfère utiliser cette mémoire si possible, à la mémoire CHIP si précieuse.

EDIT: Affiche à l'écran le nom du programme que vous êtes en train d'éditer. Cette zone sera d'abord entièrement vierge, mais lorsque vous chargez ou sauvegardez un programme sur disquette, le nouveau nom du fichier sera automatiquement entré dans la ligne d'information.

La fenêtre de l'éditeur

La fenêtre de l'éditeur est le coeur du système AMOS; elle vous permet d'entrer vos programmes Basic directement à partir du clavier. Tout texte est inséré à la **position** courante du **curseur**, ce qui est indiqué par une barre horizontale qui clignote.

Au début de chaque session, le curseur est placé dans le coin gauche du haut de la fenêtre d'édition. Vous pouvez le déplacer sur la ligne où il se trouve à présent en utilisant les flèches gauche et droite.

Vous pouvez éditer votre ligne, caractère par caractère, en utilisant les touches *Delete* et *backspace* (touche de retour arrière). *Delete* efface le caractère qui se trouve juste en dessous du curseur tandis que la touche *backspace* efface le caractère figurant à la gauche de ce curseur. Par exemple, tapez la ligne d'instruction suivante:

```
Print "AMOS"
```

Lorsque vous appuyez sur la touche *Return*, votre nouvelle instruction sera entrée dans l'AMOS Basic. Lorsqu'AMOS reconnaît une commande, celle-ci est immédiatement convertie en un format spécial.

Seule la première lettre de toutes les commandes Basic est en majuscule. L'instruction précédente sera ainsi affichée:

```
Print "AMOS"
```

D'une façon similaire, toutes les procédures et variables AMOS sont affichées en LETTRES MAJUSCULES. Ceci vous permet aisément de vérifier si vous avez commis une erreur dans une des instructions du programme. Par exemple, supposez que vous ayez tapé l'instruction suivante:

```
Inpit "Quel est votre nom; ";nom$
```

L'instruction serait ainsi affichée:

```
INPIT "Quel est votre nom; ";NOM$
```

Puisque INPIT est en MAJUSCULES, il est tout de suite évident que vous avez commis une erreur quelconque.

Ok, maintenant amusons-nous un peu. Déplacez le curseur sous la commande *Print* que vous avez entrée il y a quelques instants et tapez les lignes d'instructions Basic suivantes.

```
Centre "<Demo Machine à Ecrire>"
Do
   X$=Inkey$ : If X$<>""Then Print X$;
Loop
```

N'oubliez pas d'appuyer sur la touche *Return* après chaque ligne. Parcourez maintenant votre nouveau programme en utilisant les touches fléchées. Finalement, appuyez sur la touche F1 pour lancer ce programme.

La FENETRE DE L'EDITEUR va disparaître et un écran PROGRAMME indépendant se mettra en place. Le programme attend maintenant que vous tapiez des caractères à l'aide du clavier. Comme vous pouvez le constater, l'écran-programme est doté de son propre curseur. Il est entièrement indépendant de celui utilisé par l'éditeur. Vous pouvez ainsi vous amuser autant que vous voulez, sans changer la position actuelle de votre édition.

Pour suspendre le programme, appuyez sur Ctrl+C. Une ligne fine s'affiche maintenant à l'écran. Elle peut être déplacée en utilisant les flèches ascendantes et descendantes.

```
        Programme interrompu à la ligne 4
.. Loop
```

A ce moment précis, si vous appuyez sur la barre d'espacement, vous revenez sur l'éditeur. Mais puisque nous avons déjà vu l'éditeur, jetons plutôt un coup d'oeil au mode direct. Frappez la touche Escape pour mettre ce mode en place.

Une présentation au mode direct

Le mode DIRECT vous permet facilement de tester vos programmes Basic. Pour le moment, nous examinerons simplement deux ou trois fonctions plus intéressantes.

Vous entrez toutes les commandes en mode direct dans un écran spécial qui est complètement indépendant de l'écran-programme. Vous pouvez déplacer cet écran vers le haut ou vers le bas en utilisant les touches fléchées du clavier.

Le haut de la fenêtre contient une liste des rôles des 20 touches de fonction. Ce sont des commandes qui ont été préalablement assignées aux diverses touches de fonction. Elles sont accessibles en frappant les touches Amiga de gauche ou de droite et l'une des diverses touches de fonction.

Lorsque vous vous trouvez en mode direct, vous pouvez exécuter toutes les instructions Basic que vous voulez. Les boucles et les procédures demeurent les seules exceptions. Comme avec l'éditeur, vous devriez entrer toutes les commandes dans l'ordinateur et les valider en appuyant sur la touche Return. Voici quelques exemples:

Print 42	Imprimer une constante
ANSWER=6 : Print ANSWER*9	Effectuer un calcul
Curs Off	Mettre le curseur du texte hors fonction
Close	WorkbenchDésactiver l'utilitaire. Economise environ 40K
Run	Lancer de nouveau le programme.

Ce que vous faites en mode direct importe peu; il n'y aura absolument aucun effet sur le listing du programme en cours. Vous pouvez donc vous amuser à coeur joie, sans risquer d'effacer les données de votre programme Basic.

Il est temps de revenir à la fenêtre de l'éditeur. Faites donc vos tendres adieux au mode direct et lancez l'éditeur en appuyant sur Escape.

Notez que le curseur clignote toujours à la même position. Ceci prouve donc que les deux modes sont complètement indépendants. Faites donc un test en vous replongeant de nouveau dans le mode direct à l'aide de la touche *Escape*. Une fois contenté du mode direct, sélectionnez la fenêtre de l'éditeur. Tout doit être remis exactement dans l'état où vous l'avez laissé.

Charger un programme

Nous allons étudier les diverses procédures de chargement et de sauvegarde de vos programmes sur disquette. Vous pouvez exécuter ces options comme les autres, soit à partir de la fenêtre MENU soit en utilisant une commande simple à deux touches à partir de l'éditeur. La façon la plus rapide de charger un programme est de maintenir une des touches Amiga abaissée et d'appuyer sur la lettre L.

La classique fenêtre sélecteur de fichier AMOS vous est ensuite présentée. Les sélecteurs de fichiers se trouvent aujourd'hui dans la plupart des progiciels proposés sur l'Amiga. Si vous en avez déjà utilisé un, le système AMOS ne vous prendra pas au dépourvu.

Cependant, puisque le sélecteur de fichier est tellement inhérent au Basic AMOS, cela vaut bien la peine de l'expliquer en détails.

Le sélecteur de fichier AMOS

Sélectionnez un dossier de la disquette ne pourrait pas être plus facile. Il vous suffit de déplacer la flèche de la souris sur le fichier que vous demandez pour le mettre en vidéo inverse. Pour charger ce fichier en mémoire, cliquez deux fois le bouton gauche de la souris. Vous pouvez également taper directement son nom au clavier et appuyer simplement sur la touche Retour.

Si vous avez commis une erreur et souhaitez donc quitter le sélecteur sans charger un fichier, déplacez la souris sur le "bouton" Sortie et sélectionnez-le avec le bouton gauche. AMOS suspendra l'exécution et affichera un message "Non effectué" sur la ligne d'information.

A titre de démonstration, placez votre COPIE de la disquette programme AMOS dans le lecteur interne et appuyez sur Amiga+L pour charger un fichier. Si vous avez suivi votre guide d'initiation, AMOS vous donnera l'option de sauvegarder d'abord le programme existant. A moins que vous n'ayez effectué des changements intéressants, appuyez sur "N" pour lancer le sélecteur de fichier. Sinon, reportez-vous au Chapitre "Sauvegarder un programme" pour connaître les autres instructions.

Lorsque le sélecteur de fichier apparaît, cherchez un fichier au nom "Salut.AMOS". Une fois que vous l'avez trouvé, positionnez la souris sur le nom, et cliquez deux fois sur le bouton de gauche. Le fichier de démonstration se chargera donc en mémoire et le listing suivant se chargera en AMOS Basic.

```
Rem Salut, utilisateurs d'AMOS
Cls 0 : Rem Effacer l'écan avec couleur zéro
Do
   Rem Choisir quelques chiffres au hasard
   X=Rnd(320):Y=Rnd(200):I=Rnd(15):P=Rnd(15)
   Ink I,P : Rem Ajouter un peu de couleur
   Text X,Y,"Salut!" : Rem Texte graphique
Loop
```

Déplacez le curseur du texte sur le texte "Salut" et tapez votre propre message. Appuyez maintenant sur la touche F1 pour lancer le programme. L'écran-programme se remplira rapidement de douzaines de copies de votre texte. Appuyez sur Ctrl+C pour sortir de cette routine.

Sauvegarder un programme

Retournez à la fenêtre de l'éditeur, et tapez Shift+F3 pour sauvegarder votre programme sur disquette. Si vous aimez le changement, maintenez la touche droite de la souris abaissée et cliquez sur l'option *Sauvegarder* du menu SYSTEM à l'aide du bouton gauche. Vous revenez directement à la fenêtre du sélecteur de fichier AMOS des deux façons.

Entrez maintenant le nom de votre nouveau fichier au clavier. A mesure que vous tapez, vos lettres apparaîssent dans une petite fenêtre se trouvant au bas du sélecteur. Comme avec l'éditeur, vous pouvez voir un curseur qui se déplace au fur et à mesure que vous frappez. Ce curseur peut être déplacé en utilisant toutes les touches ordinaires d'édition. Finalement, appuyez sur la touche Retour pour sauvegarder votre programme sur disquette. Nous vous avons dit que c'était facile...

Parcourir vos fichiers

S'il n'y a presque plus de place sur votre disquette, la fenêtre de sélection ne pourra pas présenter la liste de tout le contenu de la disquette en une seule fois. Vous pouvez dérouler la liste en utilisant la barre de défilement se trouvant à la gauche de la fenêtre de sélection.

Placez la flèche de la souris sur cette barre et maintenez le bouton de gauche abaissé. Vous pouvez alors faire monter ou descendre la barre à l'aide de la souris, en déplaçant la fenêtre au fur et à mesure que vous avancez. Vous pouvez également produire un effet similaire en cliquant les icônes fléchés.

Changer le lecteur de disquettes en cours

Si vous cliquez sur le bouton droit de la souris lorsqu'un sélecteur de fichier est affiché, une liste de noms de périphérique remplacera le répertoire courant de fichier. Vous pouvez alors accéder à chaque périphérique en cliquant simplement sur celui de votre choix. AMOS listera alors tous les fichiers disponibles à partir de ce périphérique.

Le contenu précis de la fenêtre dépendra bien évidemment des périphériques branchés sur votre Amiga.

Changer de répertoire

Si vous faites une recherche dans la liste du répertoire, vous rencontrerez plusieurs noms précédés d'un astérisque (*). Ceux-ci ne sont pas des fichiers mais des sous-répertoires.

Vous pouvez lancer un de ces dossiers en les sélectionnant à l'aide du bouton gauche de la souris. Vous pouvez ensuite choisir vos fichiers directement à partir de ce dossier. Notez que seuls les fichiers possédant le suffixe courant "AMOS" seront affichés.

Une fois que vous avez ouvert un répertoire, vous pouvez le fixer en utilisant le bouton SETDIR. Si vous lancez ensuite le sélecteur de fichier ou demandez un la liste du répertoire avec DIR, le dossier que vous avez choisi sera automatiquement saisi. Similairement, vous pouvez revenir sur le répertoire précédent en cliquant sur le "bouton" PERE. C'est le petit bouton ou icône en forme de cercle se trouvant juste au-dessus de la flèche ascendante. Voir la commande PERE pour de plus amples informations.

Trier le répertoire

Tout en lisant un répertoire à partir d'un périphérique, AMOS vous permet également de sélectionner un fichier à tout moment. Cette méthode est bien plus rapide à utiliser mais a l'effet conséquent de ne pas trier le répertoire. AMOS a donc prévu un puissant bouton de TRI avec un sélecteur de fichier pour que vous puissiez trouver un fichier rapidement.

Déterminer le chemin de recherche

AMOS recherche normalement tous les noms de fichiers possédant le suffixe "AMOS". Si vous voulez charger un fichier doté d'un autre suffixe tel que .BAK, vous pouvez éditer directement le type de chemin de recherche. Opérez alors de la façon suivante:

Déplacez le curseur du texte sur la fenêtre PATH en appuyant sur la flèche verticale de votre clavier. Tapez ensuite votre nouveau chemin et frappez la touche Retour. Vous pouvez trouver une description complète de la syntaxe dans le chapitre de la commande DIR.

Attention! AMOS utilise ses propres caractères joker qui diffèrent beaucoup du système standard Amiga Dos. Si vous avez des doutes, effacez toute la ligne jusqu'aux noms courants VOLUME ou DRIVE et frappez la touche Retour. Une liste complète de TOUS les fichiers existants sur cette disquette s'affichera à l'écran.

Utiliser le sélecteur de fichiers

Ce qui est très intéressant, c'est qu'il est également possible d'appeler ce sélecteur de fichiers directement depuis vos propres programmes. Pour vous le démontrer, passez en mode DIRECT (avec Esc) et tapez l'instruction suivante:

```
Print Fsel$("*.*")
```

Le nom du fichier que vous avez sélectionné sera directement imprimé à l'écran. Voir FSEL$ pour une explication détaillée de cette commande.

Guide d'initiation de l'éditeur

Nous allons maintenant jeter un coup d'oeil à quelques-unes des fonctions d'édition les plus avancées proposées par l'éditeur AMOS. Nous allons commencer par charger un programme de démonstration d'une disquette. Pour vous mettre au défi, nous l'avons placé dans un dossier MANUEL indépendant se trouvant sur la disquette Programme AMOS.

Introduisez votre COPIE de la disquette programme dans le lecteur interne de votre Amiga et lancez la fenêtre sélecteur de fichier en appuyant sur Amiga+L. Ouvrez maintenant le dossier MANUEL en le sélectionnant à l'aide du bouton gauche de la souris. Une nouvelle liste de dossiers sera affichée dans la fenêtre de sélection de fichiers. Comme vous pouvez le constater, il y a un dossier pour chaque chapitre de ce manuel.

Pour présenter la liste des fichiers existants dans le présent chapitre, ouvrez le dossier "Chapitre 3". Il vous suffit de placer le curseur sur le nom et de cliquer sur le bouton gauche. Tous les fichiers de démonstration de ce chapitre se présentent maintenant à vous.

Enfin, chargez le fichier **Exemple 3.1.AMOS** en mémoire en le sélectionnant à l'aide de la souris. Au bout de quelques moments, le programme se chargera en AMOS Basic. Au cas où vous auriez des doutes, c'est un petit programme qui affiche à l'écran une boîte de dialogue de travail. Frappez la touche F1 si vous souhaitez une rapide démonstration. Si vous cliquez un des boutons, le programme affiche simplement son numéro et revient au Basic.

Lorsque vous examinez ce programme, vous découvrez deux ou trois faits importants. Tout d'abord, il n'y a pas un numéro de ligne en vue. En raison de la puissance de l'AMOS Basic, les numéros de ligne ne sont pas vraiment nécessaires. Nous les avons donc rendus complètement optionnels dans nos programmes Basic. C'est à vous de voir si vous souhaitez les utiliser ou non.

Une autre fonction intéressante est qu'il semble y avoir beaucoup de lignes commençant par une étrange instruction appelée "Procédure". Celle-ci constitue le point de départ de toutes les définitions de procédure de ce programme.

Les procédures sont des sections de programmes indépendantes dotées de leurs propres listes de variables et d'instructions de données. Elles sont plutôt similaires au GOSUBs que vous rencontrez dans le Basic normal. Cependant, elles sont **bien plus** puissantes! Nous allons les étudier en détail dans le Chapitre 4.

Vous pouvez étudier ce programme de diverses façons:

Parcourir un programme

Aux côtés de la fenêtre principale de l'éditeur se trouvent deux "barres de défilement". Avec l'aide de la souris, celles-ci vous permettent de parcourir votre listing.

Déplacez la flèche de la souris sur la barre Verticale et maintenez le bouton de gauche abaissé. Abaissez maintenant la barre. La fenêtre de l'éditeur défilera lentement vers le bas au travers du listing.

Vous pouvez également parcourir le programme en utilisant les icônes fléchés se trouvant en haut et en bas de cette barre. En cliquant sur ces icônes, vous déplacez la ligne d'une position dans la direction voulue.

Tout en bas de la fenêtre de l'éditeur se trouve une barre de défilement horizontale. Vous pouvez l'utiliser pour déplacer la fenêtre vers la gauche ou la droite, exactement de la même façon.

Si vous préférez utiliser le clavier pour votre édition, vous aurez également le choix entre des douzaines d'options équivalentes du clavier. Essayez celles qui suivent:

• Flèche ascendante (du clavier)	Fait monter la fenêtre d'une ligne
• Flèche descendante	Fait descendre la fenêtre d'une ligne
• Ctrl+ Flèche ascendante	Déplace le listing à la page précédente
• Ctrl+ Flèche descendante	Déplace le listing à la page suivante.

Toutes les options du clavier obéissent aux même principes de base.

Ainsi, une fois que vous vous êtes familiarisé à une commande, vous n'aurez pas de problèmes avec le reste. Vous trouverez une liste complète de ces commandes à la fin de ce chapitre.

Puisque nous avons un bon aperçu de ce programme, il est temps de passer à autre chose. Faites une recherche du listing du programme pour trouver l'instruction suivante:

```
ALERT[50,"Alert box","","Ok","Cancel",1,2]
```

Cette instruction appelle une procédure en Basic qui affiche à l'écran une boîte de dialogue. La composition de cette procédure est la suivante:

```
ALERT[Y coord,Title1$,Title2$,Button1$,Button$,Paper,Ink]
```

Transformons cette invite en quelque chose d'un peu plus passionnant. Déplacez le

curseur sur l'instruction ci-dessus et éditez la ligne à l'aide des touches de déplacement du curseur afin qu'elle ressemble à ce qui suit:

```
ALERT(50,"Massacrer!","Stephen","Ouais!","Ah non!",13)
```

Exécutez le programme en appuyant sur la touche F1 ou en sélectionnant DEROULER du menu principal. Il vous sera donné l'unique option d'arrêter l'auteur du manuel sur sa piste. Sélectionnez un bouton à l'aide de la souris et faites votre choix. Aïe! a fait mal!

En fait, vous pouvez vraiment changer le titre et les "boutons" autant que vous voulez. N'hésitez pas à utiliser cette routine dans vos propres programmes.

Avec un peu de chance, l'exemple ci-dessus vous aura incité à utiliser ces procédures dans vos propres programmes. Afin de vous aider dans cette tâche, nous avons élaboré un grand nombre de fonctions d'édition spéciales dans l'éditeur AMOS.

Recherches d'étiquettes et de procédures

Si votre programme est très long, vous pouvez trouver difficile de déceler les points de départ des diverses définitions de procédure. Nous avons donc prévu la possibilité de sauter directement à la définition de la procédure suivante au moyen de deux touches (Alt+Flèche).

Pour exécuter cette fonction, placez le curseur au début du programme et appuyez sur Alt+flèche descendante. Votre curseur sera immédiatement déplacé sur le premier caractère de la première définition de procédure du programme courant (ALERT). Vous pouvez répéter ce processus pour sauter d'une définition de procédure à l'autre. Une fois que vous êtes arrivé à la fin du listing, vous pouvez remonter de la même façon en utilisant Alt+flèche ascendante .

Bien évidemment, ce système n'est pas simplement limité aux procédures. Il fonctionne aussi bien avec les étiquettes qu'avec les numéros de ligne. Par conséquent, même si vous n'avez pas besoin de procédures, vous trouverez toujours cette fonction utile.

Replier une définition de procédure

Si vous élaborez vos programmes à partir d'un bloc de procédures très souvent utilisées, vos listings peuvent facilement être encombrés de définitions de diverses routines de bibliothèque.

Heureusement, on peut venir à votre secours. Avec un simple appel à la commande *Replier*, vous pouvez cacher n'importe quelle de vos définitions de procédure venant de vos programmes. Comme les autres, ces routines peuvent être utilisées dans votre programme, mais leurs définitions sont remplacées par une seule instruction de Procédure. Par exemple:

Positionnez le curseur dans la définition de ALERT et cliquez sur l'option *Replier/Déplier* de la fenêtre menu. Et hop! Le contenu de votre procédure se volatilise! Mise à part ceci, vous pouvez lancer votre programme sans effets secondaires. Seule l'apparence du listing dans la fenêtre de l'éditeur se trouve modifiée.

Si vous voulez modifier cette procédure, il est assez facile de revenir au listing d'origine. Il vous suffit de resélectionner *Replier/Déplier* et votre procédure sera entièrement redéveloppée.

Il est également possible de replier TOUTES les procédures de votre programme à la fois. Pour cela, l'option du menu CHERCHE appelée *Verrouillage complet* doit être sélectionnée. Cliquez sur le "bouton" menu CHERCHE pour amener ce menu à l'écran, ou appuyez sur la touche F5 du clavier. Sélectionnez maintenant le "bouton" *PlierTT* pour retirer les définitions de procédure du programme courant.

L'effet produit sur le programme **EXEMPLE 3.1** est impressionnant! Le programme entier entre maintenant dans un simple écran. Vous pouvez ainsi voir immédiatement quelles procédures nous avons utilisées dans le programme. Vous pouvez éditer chaque définition de procédure en la développant grâce au "bouton" *Replier/Déplier.* Vous pouvez également déplier tout le programme en sélectionnant Ouvrir Tout à partir du menu CHERCHE.

Chercher/Remplacer

Vous pouvez accéder aux commandes *Chercher/Remplacer* proposées par l'éditeur AMOS Basic au moyen d'un menu RECHERCHE spécial qui peut être appelé soit à partir de la fenêtre du menu, soit en appuyant sur la touche de fonction F4.

Trouver une expression

Nous poursuivons l'étude de notre guide d'initiation en examinant rapidement quelques-unes des instructions *Chercher/Remplacer.* Commençons par la commande *Chercher.*

Vous pouvez l'exécuter soit directement à partir du menu CHERCHE, soit en utilisant les touches Ctrl+F du clavier. Une fois que vous avez sélectionné cette commande, il vous sera demandé de frapper la chaîne recherchée.

Par exemple, appuyez sur Ctrl+F et tapez Rem à l'apparition de l'invite. AMOS va maintenant chercher le message *Rem* suivant dans votre programme, en partant de la position initiale du curseur. Si la recherche aboutit, le curseur sera placé sur l'expression recherchée.

Vous pouvez de nouveau répéter la recherche à partir de cette nouvelle position du curseur en sélectionnant l'option *Suivant* (Ctrl+N).

Remplacer

Supposez que vous voulez changer tous les *Rem* d'un programme par les caractères équivalents suivants " ". Vous pouvez accomplir ceci en utilisant la *commande Remplacer.*

Pour cela, il vous faut définir la chaîne de remplacement. Ainsi, la prochaine fois que vous appelez *Remplacer,* il vous sera demandé de frapper cette chaîne en utilisant le clavier.

Appuyez sur Ctrl+R, tapez une apostrophe " ' " à l'apparition de l'invite et frappez la touche retour pour l'entrer dans l'ordinateur. Vous pouvez maintenant entrer la chaîne recherchée en sélectionnant l'option *Chercher* de la manière suivante:

- Appuyez sur Cnrl+F pour sélectionner l'option *Chercher*
- Tapez *Rem* dans la ligne d'information
- Le curseur se déplacera directement sur le prochain *Rem* du listing de votre programme.

Si vous souhaitez procéder au remplacement de la chaîne et faire un saut à la prochaine occurence, sélectionnez de nouveau *Remplacer* (Ctrl+R). Si le *Rem* se trouve au milieu de la ligne, il vous faut le contourner parce qu'AMOS vous permet seulement de substituer un libellé à cette commande au début d'une ligne. Vous pouvez éviter ce problème et faire un saut direct à la prochaine expression de votre programme en utilisant *Suivant*.

Couper/Coller

Les commandes Bloc AMOS vous permettent de découper vos programmes et de sauvegarder chaque section en mémoire pour les utiliser ultérieurement. Une fois que vous avez créé un bloc, vous pouvez le copier à l'endroit où vous voulez dans le programme courant.

Voici un exemple d'application de cette fonction. Prenons le programme ALERT précédent et découpons une seule procédure. Plaçons la flèche de la souris sur la première ligne de la procédure INVERSER et enfoncons le bouton droit de la souris. Nous pouvons maintenant faire entrer cette procédure dans un bloc à l'aide de la souris. Maintenez la touche droite de la souris abaissée et déplacez la flèche vers le bas de l'écran. A mesure que vous bougez la souris, la zone sélectionnée sera mise en vidéo inverse.

Nous pouvons maintenant saisir cette zone et la mettre en mémoire en utilisant *Couper*. Si vous appuyez sur les touches Ctrl+C du clavier, la procédure sera retirée du listing et mise en mémoire. Il est maintenant possible de coller ce bloc à n'importe quel endroit de votre programme. A titre d'application, déplacez le curseur du texte en fin de programme et appelez l'option *Coller* en appuyant sur Ctrl+P. La procédure INVERSER sera copiée à la position courante du curseur.

Programmes multiples et acessoires

Programmes multiples

Bien qu'AMOS ne vous permette d'éditer qu'un programme à la fois, le nombre de programmes que vous pouvez mettre en mémoire est illimité et dépend seulement de la capacité de votre mémoire. Une fois que vous avez mis un programme en place de cette façon, vous pouvez l'exécuter directement depuis la fenêtre de l'éditeur à l'aide de l'option *Run Autre*.

Si votre système dispose d'une mémoire d'extension, vous pourrez facilement conserver en mémoire deux ou trois programmes de grande taille sans problème. Vous trouverez cette fonction utile même si vous utilisez seulement un Amiga 500 standard doté d'une mémoire de 512k.

Par exemple, supposez que vous rencontriez un problème dans le lancement de l'un de vos programmes. AMOS vous permet facilement d'échanger ce programme contre celui qui est existant en mémoire afin que vous puissiez librement mettre en pratique les diverses possibilités offertes jusqu'à ce que vous trouviez une solution. Vous pouvez ensuite saisir votre nouvelle routine en mémoire à l'aide de l'option Couper et il vous suffit d'appuyer sur deux touches pour revenir sur votre programme d'origine! Vous pouvez alors coller la nouvelle routine et continuer votre programme comme auparavant. Cette capacité d'arrêter tout et d'essayer ses idées immédiatement est incroyablement précieuse.

Il vous est également possible de garder en permanence tous les programmes utilitaires les plus utilisés tels que l'éditeur de sprite ou l'éditeur de carte en mémoire. Vous pouvez avoir accès à ces utilitaires en un instant et à tout moment.

En fait, AMOS comprend un système ACCESSOIRE spécial qui facilite encore davantage cet accès. Lorsque vous vous trouvez dans vos principaux programmes, vous pouvez permettre aux programmes utilitaires d'accéder librement aux banques de mémoire. Ainsi l'éditeur de sprites peut s'emparer directement des images de votre programme courant et les modifier immédiatement. Cette technique accélère le processus global de mise au point à une vitesse exceptionnelle!

Faisons une rapide démonstration de ces fonctions. Lancez le petit programme suivant dans l'éditeur:

```
Print "Voici programme un"
Boom
```

Nous pouvons maintenant "pousser" ce programme en mémoire en utilisant la commande *pousser.* Vous pouvez l'appeler en appuyant sur Amiga+P du clavier. Il vous sera ensuite demandé de taper le nom de votre programme sur la ligne d'information. Tapez par exemple le nom "Programme 1". L'écran d'édition s'effacera entièrement.
La nouvelle fenêtre est complètement indépendante de votre programme d'origine. A titre de démonstration, lancez une deuxième routine en tapant les lignes suivantes:

```
Print "Voici programme deux"
Shoot
```

Vous pouvez maintenant exécuter ce programme depuis la fenêtre de l'éditeur en utilisant RUN (F1). Mais lorsque votre programme revient, vous pouvez immédiatement faire un saut au précédent en utilisant l'option *Flip.*

Appuyez sur sur les touches Amiga+F du clavier. Il vous sera encore demandé de taper le nom de votre programme. Choisissez le nom "Programme 2" à cet effet. L'éditeur reviendra d'un saut à votre premier programme comme par magie.

Il est possible de répéter ce processus d'échange de deux programmes. Chaque programme est complètement indépendant et peut avoir sa propre liste de banques et d'écrans-programmes.

Jusqu'à maintenant, nous avons seulement vu comment nous pouvons utiliser simultanément deux programmes. En fait, vous pouvez avoir autant de programmes en mémoire que vous souhaitez. Vous pouvez sélectionner chaque programme en utilisant les options *Run Autre* et *Editer Autre* de la fenêtre Menu. Lorsque vous appelez ces commandes, un sélecteur spécial "programme" s'affiche à l'écran.

Le sélecteur de programmes est presque identique au sélecteur de fichier AMOS que vous connaissez. La seule différence est qu'il vous permet de choisir un programme en mémoire et non sur le disque. Pour sélectionner un programme, il vous suffit de le mettre en surbrillance à l'aide de la flèche du curseur et de cliquer une seule fois le bouton gauche.

Essayez de lancer et d'éditer le Programme 1 et le Programme 2 en utilisant ce système. Une fois que vous avez compris, c'est extrêmement simple à utiliser. Voir les commandes *Charger Autre* et *Effacer Autre.*

Les accessoires

Afin de faire la distinction entre les accessoires des programmes Basic ordinaires, nous leur avons affecté un suffixe "ACC" au lieu du classique "AMOS". Les accessoires peuvent être chargés en mémoire comme tout programme ordinaire en utilisant la commande *Charger Autre.*

Cette commande vous présente un sélecteur de fichier ordinaire qui peut servir à charger un programme accessoire de la disquette. Après avoir installé l'accessoire en mémoire, vous êtes renvoyé directement à votre programme courant. Vous pouvez alors lancer cet accessoire à tout moment en utilisant l'option Run Autre de la fenêtre Menu. Il vous suffit de déplacer la flèche de la souris sur l'accessoire désiré et d'appuyer sur le bouton gauche.

Vous pouvez également charger tous les accessoires du disque courant en utilisant la fonction *AccNouv/Charg.* Vous pouvez sélectionner cette option du menu Système; celui-ci s'affiche si vous maintenez le bouton droit de la souris abaissé. *AccNouv/Charg* efface tous les accessoires existants et charge un nouveau jeu depuis le disque courant.

A titre de démonstration, placez le disque programme AMOS dans votre lecteur de disquettes et cliquez sur le bouton *AccNouv/Charg* du menu système.

L'accessoire HELP sera rapidement chargé dans la mémoire. HELP est un accessoire spécial parce qu'il peut être appelé directement en appuyant sur la touche HELP du clavier. Toutes les informations dont vous avez besoin à propos des programmes accessoires livrés avec l'AMOS Basic accompagnent ce programme. Il vous suffit donc de suivre les invites apparaîssant à l'écran.

Le mode direct

Vous pouvez lancer la fenêtre du mode direct depuis l'éditeur en appuyant sur la touche ESCape à votre convenance. La fenêtre sera affichée par défaut dans la moitié inférieure de l'écran avec l'écran- programme en arrière-plan.

Si vous exécutez un programme qui change le format de l'écran, affiche les fenêtres, anime les sprites, etc..., toutes ces données à l'écran resteront intactes. Vous pouvez donc déplacer la fenêtre du mode DIRECT ou revenir à l'éditeur pour effectuer des changements de programmes sans détruire l'écran-programme courant. La fenêtre en mode DIRECT est complètement indépendante et s'affiche en premier plan.

Lorsque vous êtes en mode direct, vous pouvez taper n'importe quelle ligne en AMOS Basic que vous souhaitez. Les seules commandes que vous ne pouvez pas utiliser sont les boucles et les instructions de branchement. Vous n'avez accès qu'aux variables ordinaires (différentes des variables locales définies dans une procédure).

Les touches de l'éditeur en mode direct

ESCape	Fait un saut à la fenêtre de l'éditeur.
Retour	Exécute la ligne courante de commandes.
DELete	Efface les caractères se trouvant en dessous du curseur.
RAPPEL ARRIERE	Efface les caractères se trouvant à la gauche du curseur.
Flèche gauche	Déplace le curseur vers la gauche.
Flèche droite	Déplace le curseur vers la droite.
Shift+gauche	Saute d'un mot à l'autre vers la gauche.

Shift+droite	Saute d'un mot à l'autre vers la droite.
Shift+RETOUR	Efface la ligne
Help	Affiche les définitions de la touche de fonction dans la fenêtre direct.
de F1 à F10:	Ces touches vous rappellent les 10 dernières lignes que vous tapées en mode direct. F1 affiche la toute dernière. F2 la seconde et ainsi de suite jusqu'à la dernière. La zone mémoire utilisée dans ce système est toujours effacée lorsque vous revenez à la fenêtre de l'éditeur ou lorsque vous exécutez un de vos programmes.

La fenêtre menu

Voici une explication détaillée de toutes les options qui vous sont proposées à la fenêtre du menu principal.

Le menu implicite

Il liste les diverses commandes vous permettant de faire fonctionner l'éditeur; en outre, il vous permet d'accéder aux menus *bloc* et *recherche.*

Run **(F1)**
Exécute le programme courant à partir de la mémoire.

Tester **(F2)**
Vérifie la syntaxe de la totalité du programme et place le curseur à la première erreur.

Indenter **(F3)**
Saisit le programme courant et décale proprement le listing pour vous.

Menu Blocs **(Ctrl ou F4)**
Affiche le menu *Bloc* dans la fenêtre de sélection. Ces options peuvent maintenant être appelées soit à l'aide de la souris soit en appuyant sur la touche de fonction appropriée du clavier. Vous pouvez revenir au menu principal en cliquant sur le bouton droit de la souris et en déplaçant la flèche en dehors de la zone de la touche de fonction ou en frappant une touche.

Recherche **(Alt ou F5)**
Amène le *Menu Recherche* à l'écran. Il vous permet de faire une recherche de mots clés spécifiques et de les remplacer au besoin. Vous pouvez également régler la taille du texte-tampon ou changer la tabulation courante depuis le menu.

Run Autre **(F6)**
Exécute un programme ou un accessoire détenu dans la mémoire de l'Amiga.

Edite Autre **(F7)**
Edite un programme qui a été préalablement mis en mémoire en utilisant la commande *Charger Autre* ou la commande *AccNouv/Charg.* Si vous n'avez pas sauvegardé votre

programme courant, vous serez invité à donner son nom.

Votre programme courant sera alors *poussé* en mémoire et vous pourrez choisir un autre programme pour l'édition en utilisant le sélecteur du programme. Pour sélectionner ce programme, il vous suffit de déplacer la souris sur son nom et d'appuyer sur le bouton gauche.

Notez que la taille de *l'éditeur-tampon* est mémorisé avec votre programme. Cette zone-tampon reprendra sa taille initiale lorsque vous rééditerez votre programme. Si vous essayez d'éditer un programme, tel qu'un accessoire, qui n'a pas été sauvegardé de cette façon, la capacité de la mémoire-tampon sera, au besoin, automatiquement augmentée . Si vous êtes à cours de mémoire, l'option sera suspendue en faisant apparaître un message d'erreur *"Mémoire insuffisante"*.

Remplacmt (F8)

Alterne deux mode d'édition indépendants.

Mode insertion: (mode par défaut), insère dans la ligne chaque caractère tapé. Si vous étiez en train d'éditer la ligne suivante:

Rem TESTER LE MODE INSRTION

Puisque le curseur se trouve en dessous du R, taper un "E" modifierait ainsi la ligne:

Rem TESTER LE MODE INSERTION

Mode Superposition: En mode *Superposition*, l'appui d'une touche permet de remplacer le caractère se trouvant sous le curseur. En prenant l'exemple précédent, taper un "E" changerait de nouveau la ligne comme suit:

Rem TESTER LE MODE INSETION

Après avoir changé de mode, l'article du menu sera positionné sur **Superposition.** En sélectionnant cette option, vous revenez sur le système normal d'édition. Le mode de frappe en cours est indiqué par une lettre se trouvant à l'extrême gauche de la ligne d'information. I=INSERTION et S=SUPERPOSITION. Notez que si vous commettez une erreur lorsque vous êtes en mode *Insertion*, vous pouvez normalement inverser les modifications sur la ligne courante en appuyant sur Ctrl+U.

Ouvre/Ferme (F9)

Saisit une définition de procédure et la range dans le listing de votre programme. Lorsque vous avez plié une procédure, seule sa première ligne sera affichée dans vos listings.

Pour plier une procédure, déplacez le curseur dans la procédure et sélectionnez l'option *Replier*. Les erreurs de syntaxe seront recherchées dans votre programme et la procédure sera cachée dans votre programme. Il est très important de comprendre qu'il ne s'est absolument **rien** passé dans les lignes de programmes de la procédure. Seule la façon dont ces lignes sont affichées est différente.

Il est normalement possible de réouvrir une procédure qui a été pliée en répétant ce processus. Placez le curseur sur une procédure pliée et cliquez sur *Replier/Déplier*.

(Frappez F9 qui est un raccourci- clavier.) S'il vous faut davantage de sécurité, vous pouvez également appeler un accessoire spécial VERRROUILLAGE du disque programme AMOS.

- Faites une sauvegarde de votre programme avec des procédures **ouvertes**
- Fermez toutes les procédures que vous voulez absolument verrouiller
- Chargez le programme FOLD.AMOS en tant qu'ACCESSOIRE
- Faites fonctionner FOLD.AMOS en utilisant l'option du menu RUN AUTRE de l'éditeur

Le véritable attrait de ce système est qu'il vous permet de créer des bibliothèques entières de vos routines sur disquette. Vous pouvez charger ces dernières en mémoire en tant que programme indépendant (voir CHARGER AUTRE). Vous pouvez maintenant découper la routine dont vous avez besoin et la copier directement dans votre programme principal. Ainsi, une fois que vous avez écrit une routine, vous pouvez la placer dans une procédure et l'utiliser autant de fois que vous le souhaitez.

Si vous avez l'intention d'utiliser ce système, il vous faut examiner les différents points suivants:

- Lorsque vous pliez ou dépliez une procédure, une vérification de syntaxe est effectuée dans **l'intégralité** du programme. Si une erreur se produit, l'exécution ne se fera pas. Il est donc très important de conserver des copies de sauvegarde de toutes vos procédures dans un format déplié.
- N'essayez pas d'effacer une procédure pliée en utilisant les touches habituelles de déplacement du curseur. Elle ne sera pas effacée de façon permanente. Il vous faut donc délimiter votre procédure en utilisant COUPER.
- Le Couper/Coller marche bien avec les procédures pliées. Toute la définition de procédure est mise en mémoire lorsque vous découpez le libellé de la Procédure. Mais vous devriez faire attention d'éviter les erreurs suivantes.

"Ce tableau n'est pas défini dans votre programme principal"

Lorsque vous copiez une routine d'un programme à l'autre, il est facile d'oublier les variables SHARED ou GLOBAL et les tableaux que vous avez définis dans votre programme principal. Si vous utilisez des variables externes pour une certaine procédure et celles-ci n'ont pas été définies, vous obtiendrez le message d'erreur ci-dessus. Recherchez donc les libellés SHARED ou GLOBAL dans votre programme original.

"Etiquette définie deux fois"

Vous avez essayé de faire DEUX copies d'une procédure dans le même programme! Vous vous êtes sans doute emparé d'une procédure supplémentaire par erreur.

"Procédure non définie"

En AMOS Basic, vous pouvez très bien appeler les procédures l'une dans l'autre. Ceci cause occasionnellement des problèmes lorsque vous essayez de sortir de l'une de ces

procédures, puisque l'exécution se fait seulement si les procédures utilisées ont été également copiées dans votre programme. Il est bon de lister ces routines dès le début d'une procédure, car ceci peut éviter **beaucoup** de confusion!

Ins. Ligne **(Ctrl+I ou F10)**
Insère une ligne à la position actuelle du curseur.

Le menu SYSTEM

Le menu SYSTEM comprend une série de commandes vous permettant de charger et de sauvegarder vos programmes à partir de la disquette. Pour sélectionner ce menu, il vous suffit d'appuyer soit sur la touche *shift* ou de maintenir le bouton droit de la souris abaissé. Voici une liste complète des options proposées.

Charger **(Shift+F1 ou Amiga+L)**
Charge un fichier AMOS Basic à partir du disque. Ce fichier est choisi en utilisant le sélecteur de fichier standard AMOS.

Sauver **(Shift+F2 ou Amiga+S)**
Sauvegarde le programme courant AMOS. Si vous sauvegardez un fichier pour la première fois, il vous sera demandé d'entrer son nom à l'aide du sélecteur de fichier. S'il existe un autre programme portant le même nom sur le disque, il sera automatiquement renommé en ajoutant le suffixe ".BAK". Ceci constitue une protection bien pratique contre toute éventuelle erreur puisque vous pouvez normalement revenir à la version précédente de votre programme si vous avez fait une bêtise.

Sauver com. **(Shift+F3 ou Shift+Amiga+S)**
Sauvegarde le programme courant sous un autre nom. La sélection du nom se fait à l'aide d'un sélecteur de fichier. Notez que si vous sauvegardez un programme en utilisant le nom AUTOEXEC.AMOS, celui-ci sera automatiquement chargé et exécuté dès la mise en route. Lorsque vous lancez l'AMOS Basic, le principal écran-programme s'affiche immédiatement.

Fusionner **(Shift+F4)**
Lance le programe sélectionné à la position actuelle du curseur sans effacer d'abord votre programme d'origine. Cette fonction est utilisée pour intégrer les routines saisies dans d'autres programmes, telles que l'Editeur de Terrain: TAME.

Fusionner Asc. **(Shift+F5)**
Fusionne la version Ascii d'un programme en AMOS Basic et un programme existant en mémoire.

Accessories **(Shift+F6)**
Supprime toutes les routines accessoires courantes de la mémoire et en lance un nouveau jeu de la disquette. Cette commande charge automatiquement tous les fichiers comportant le suffixe ".ACC". Si vous utilisez un Amiga 500 doté d'une mémoire non-étendue, vous devriez traiter cette commande avec quelques précautions car il est très facile de se retrouver à court de mémoire.

Charg. Autre **(Shift+F7)**
Charge un seul programme accessoire de la disquette courante. Vous pouvez l'accéder en utilisant la commande *Run Autre* du menu principal.

Eff. Autre **(Shift+F8)**
Efface un ou plusieurs accessoires de la mémoire de votre Amiga. Lorsque vous appelez cette commande, le sélecteur normal du programme AMOS se présentera à vous. Vous pouvez maintenant cliquer sur un seul programme à effacer ou sur le bouton **tout** pour supprimer tous les accessoires.

Nouveau **(Shift+F9)**
Efface le programme que vous êtes en train d'éditer. Si vous n'avez pas sauvegardé votre programme, il vous sera donné l'option de le conserver sur disquette avant qu'il soit supprimé.

Quitter **(Shift+F10)**
Sort d'AMOS et repasse la main au CLI. Comme avec NEW, il vous est donné l'option de sauvegarder votre présent programme avant de quitter AMOS.

Le menu BLOCS

Le menu BLOCS vous offre une fonction pratique qui permet de déplacer des sections entières de votre programme d'un endroit à l'autre.

Au besoin, vous pouvez accéder à ces fonctions directement du menu Principal à l'aide de la souris. Mais vous allez sans doute trouver qu'il est plus rapide d'utiliser les commandes de remplacement du clavier. Voici une liste complète des diverses options.

Debut Bloc **(Ctrl+B ou Ctrl+F1)**
Positionne le point de départ du bloc courant.

Fin Bloc **(Ctrl+E ou Ctrl+F6)**
Définit le point terminal du bloc. La fonction est normalement utilisée juste après la comande Ctrl+B. La zone comprise entre le point de départ et le point terminal est affichée en vidéo inverse.

Couper Bloc **(Ctrl+C ou Ctrl+F2)**
Retire le bloc sélectionné de sa position actuelle et la charge en mémoire. Vous pouvez maintenant copier ce bloc à n'importe quel point de votre programme en utilisant la commande *Coller*.

Copier Bloc **(Ctrl+P ou Ctrl+F7)**
Colle tout le contenu d'un bloc à la position actuelle du curseur. Vous devez d'abord sauvegarder ce bloc en mémoire en utilisant les commandes *Couper* ou *Ranger*.

Déplacer Bloc **(Ctrl+M ou Ctrl+F3)**
Déplace le bloc que vous avez mis en surbillance à la position courante du curseur en faisant disparaître entièrement le bloc d'origine.

Stocker Bl. **(Ctrl+S ou Ctrl+F8)**
Copie le contenu d'un bloc en mémoire sans affecter le programme courant. Cette option vous permet simplement de transférer des lignes d'un programme en mémoire à l'autre.

Cacher Bloc **(Ctrl+H ou Ctrl+F4)**
Enlève les crochets délimitant le bloc que vous avez mis en surbrillance en utilisant les commandes *Début Bloc* et *Fin Bloc*.

Sauver Bloc **(Ctrl+F9)**
Sauvegarde le bloc courant sur le disque en tant que programme AMOS. Vous pouvez maintenant le recharger en utilisant les commandes *Fusionner* ou *Charger* du menu SYSTEM. Notez que les banques mémoire de votre programme **ne sont pas** sauvegardés avec votre listing.

Sauver Ascii **(Ctrl+F5)**
Mémorise le bloc que vous avez sélectionné sur le disque en tant que fichier-texte. Vous pouvez recharger ce fichier directement sur n'importe quel traitement de texte standard. Si vous avez accès à un Modem, vous pouvez également charger ces fichiers sur des tableaux d'affichage et des réseaux de communication.

Imprimer **Bloc (Ctrl+F10)**
Sort le bloc sélectionné directement sur imprimante si celle-ci est branchée.

Il existe également une commande spéciale *Selection Tout* à laquelle vous pouvez seulement accéder à l'aide du clavier. En appuyant sur les touches **Ctrl+A**, vous sélectionnez en bloc la totalité du programme courant. Ceci est utile lorsqu'il vous faut créer un fichier de sauvegarde de tout le programme en Ascii.

Le menu CHERCHE

Une des meilleures façons d'étudier le système AMOS est d'examiner quelques-uns des programmes de démonstration que nous vous avons livrés avec la disquette de données AMOS. Avec l'aide du menu CHERCHE AMOS, vous pouvez rechercher des exemples d'instruction AMOS dans les listings. Cela vous donnera des indications précieuses sur la façon de l'utiliser dans le contexte d'un vrai programme.

Vous pouvez également utiliser la commande *Remplacer* pour modifier tous les noms de variables d'un de vos programmes Basic. Vous pouvez donc utiliser des noms courts, simples lorsque vous lancez vos programmes puis les convertir ensuite en quelque chose de plus lisible lorsque vous avez fini.

Vous pouvez appeler le menu CHERCHE directement de la fenêtre menu en utilisant la souris. Sinon, utilisez une des commandes abrégées du clavier pour appeler directement la fonction que vous voulez.

Rechercher **(Ctrl+F ou Alt+F1)**
Saisit une chaîne d'au plus 32 caractères et fait une recherche dans votre texte jusqu'à exacte concordance. La recherche s'effectue à partir de la position du curseur.

Suivant **(Ctrl+N ou Alt+F2)**
Recherche la prochaine occurence de la chaîne que vous avez spécifiée en utilisant *Chercher.*

Rech. Haut **(Alt+F3)**
Cette fonction est identique à *Chercher* sauf qu'elle démarre la recherche à partir du début du programme et non à la position actuelle du curseur.

Remplacer **(Alt+F4 ou Ctrl+R)**
Active le mode Remplacer. L'effet que produit cette commande varie selon son mode d'utilisation. Il existe deux variantes de ce mode:

• **Avant** une commande *Chercher*

Il vous est maintenant demandé d'entrer la chaîne de remplacement à l'aide du clavier.

• **Après** une commande *Chercher*

Si la recherche aboutit, le texte sera remplacé par la chaîne de remplacement à la position courante du curseur. Exécutez la commande *Remplacer* pour afficher la prochaine occurence de la chaîne de recherche de votre programme. Si vous ne voulez pas remplacer cet article, vous pouvez sauter directement au prochain mot avec *Rechercher* .

Remp. Tout **(Alt+F5)**
Remplace **tous** les mots identiques de votre programme. La procédure est la suivante:

- Validez la commande en frappant "Y" du clavier ou en cliquant sur la case *Oui* de la ligne d'information.
- Entrez la chaîne que vous voulez changer.
- Entrez la chaîne de remplacement. La recherche/remplacement s'exécute maintenant depuis le **début** de votre programme.

Min = Maj **(Alt+F6)**
Change la distinction implicite des minuscules et des majuscules dans vos diverses commandes *Chercher/Remplacer.* Un "g" et un "G" seraient autrement traités comme des lettres différentes dans vos recherches. Cette option force AMOS à assumer que les caractères en minuscule ou en majuscule de votre texte sont identiques. La fonction **Maj=Min** sera affichée à l'écran vous indiquant ainsi le nouveau mode d'opération.

Ouvrir Tout **(Alt+F7)**
Ouvre toutes les procédures fermées de votre programme. Une vérification de la syntaxe de l'intégralité du programme est faite avant que les procédures soient dépliées. Si une erreur est détectée, l'opération sera suspendue.

Si cette fonction vous pose des problèmes, référez-vous au chapitre traitant la commande *Replier;* il vous donnera une explication détaillée des solutions répondant à vos problèmes.

Fermer Tout **(Alt+F8)**
Ferme toutes les définitions de procédure de votre programme courant. Seule la première ligne de vos définitions de procédure sera affichée dans vos listings. Ceci les rend bien plus courtes et désemcombre bien vos listings. Comme avec la commande précédente *Ouvrir*, l'opération de pliage sera exécutée seulement si il n'y a pas d'erreurs dans votre programme courant.

Une fois que vous avez fermé vos procédures, vous pouvez les éditer individuellement en utilisant l'option indépendante *Replier/Déplier*.

Tampon Text **(Alt+F9)**
La commande FIXER TEXTE TAMPON change le nombre de caractères disponibles pour mettre vos listings en attente. Vous pouvez utiliser cette commande pour augmenter la mémoire des éditeurs ce qui vous permet ainsi d'entrer des programmes plus importants dans votre Amiga. Elle peut s'avèrer être très utile si vous avez étendu la mémoire de votre Amiga. Voir également à CLOSE EDITOR.

Tabulation **(Ctrl+TAB ou Alt+F10)**
Cette commande détermine le nombre de caractères de déplacement du curseur lorsque vous appuyez sur la touche TAB.

Les macros du clavier

AMOS Basic vous donne le choix de créer 20 macros du clavier à la fois. Vous pouvez y accéder en appuyant sur la touche gauche ou droite de l'Amiga et sur une touche de fonction. Une fois que vous avez défini une macro, vous pouvez l'utiliser dans le système AMOS, comme si vous aviez exécuté vos commandes directement depuis le clavier. Vous pouvez appeler la même macro depuis la fenêtre de l'éditeur lorsque vous êtes en mode direct, ou lorsque vous êtes dans un de vos programmes Basic!

Les affectations courantes des touches sont affichées dans un champ spécial se trouvant au-dessus de la fenêtre du mode direct. Lorsque vous êtes en mode direct, vous pouvez appeler cette liste à l'écran à l'aide de la touche HELP. De la même façon, si vous appuyez sur les touches Amiga de gauche ou de droite depuis l'éditeur, ces définitions apparaîtront dans la fenêtre menu.

Toutes les affectations macro sont implicitement chargées d'un jeu de mots clés en Basic ordinaire. Ces derniers peuvent être changés en utilisant une option simple de l'accessoire configuration (CONFIG.ACC). Il est également possible d'affecter ces touches directement lorsque vous êtes dans un de vos programmes en utilisant la puissante fonction KEY$.

=KEY$= *(Définition d'une macro du clavier)*

KEY$(n)=command$
command$=KEY$(n)

KEY$ affecte le contenu de *command$* au numéro de la touche de fonction *n*. *n* est un numéro d'identification de votre touche de fonction compris entre un et vingt.

Vous pouvez accéder aux touches dont les numéros d'identification sont compris

entre un et dix en appuyant conjointement sur la touche de fonction et sur le bouton gauche de votre Amiga. De la même façon, vous pouvez appeler les nombres supérieurs ou égaux à onze à l'aide de la combinaison bouton droit de l'Amiga et une touche de fonction.

Notez qu'il est important d'appuyer sur les deux touches simultanément, sinon votre macro sera interprétée comme l'appui de deux touches indépendantes.

command$ peut représenter n'importe quelle chaîne de texte que vous souhaitez, d'une longueur maximale de 20 caractères. Cette fonction interprète directement deux caractères spéciaux.

.(Apostrophe inversé)	Produit un code de renvoi.
.(Apostrophe)	Délimite un commentaire. Celui-ci est uniquement affiché dans vos listes clés. Il est entièrement ignoré par la routine macro. Exemples:

```
? Key$(1)
Key$(2)="Défaut'"
Alt+F2

Key$(3)="Commentaire' Print"
```

En pratique, le système macro peut s'avérer extrêmement utile. Vous pouvez non seulement accélérer le processus d'entrée de vos programmes Basic, mais vous pouvez également définir une liste de paramètres classiques pour vos programmes Basic. Comme vous pouvez le constater dans le programme **EXEMPLE 3.2** dans le dossier MANUEL, ceux-ci seraient très efficaces dans un jeu d'aventures.

Si vous souhaitez créer une frappe n'ayant pas d'équivalent en Ascii, comme une flèche ascendante, vous pouvez intégrer un code de position de touche à ces macros. Ceci est accompli par la fonction SCAN$.

=SCAN$ *(Renvoi d'un code de position utilisé avec KEY$)*

x$=Scan$(n [,m])

n est le code de position de la touche utilisé dans une des définitions macro. *m* est un masque optionnel fixant les touches spéciales telles que Ctrl ou Alt dont le format est le suivant:

Bit	Touche Testée	Notes
0	Touche SHIFT de gauche	
1	Touche SHIFT de droite	
2	Mode Majuscules	Soit ACTIVE ou DESACTIVE
3	Contrôle (Ctrl)	
4	Alt de gauche	
5	Alt de droite	
6	Amiga de gauche	Touche Commodore sur certains claviers
7	Amiga de droite	

Si un bit est forcé à un, alors le bouton qui lui est associé est "enfoncé" dans votre macro. Exemples:

```
KEY$(4)="Super!"+Scan$($4C)
KEY$(5)="Page d'en haut!"+scan$($4c,%00010000)
```

Conserver la mémoire

Si vous utilisez un Amiga 500 doté d'une mémoire non-étendue, il se peut que vous manquez quelquefois de place. Nous avons donc prévu deux instructions d'une portée sur-puissante qui vous permettront de maximiser la mémoire disponible pour vos programmes.

Avant d'étudier ces fonctions, il est bon de noter que si vous disposez d'un lecteur externe de disquettes de 3,5 pouces, vous pouvez économiser environ 30k en le désactivant avant de charger AMOS Basic. Puisque AMOS n'accède le disque que pour quelques petits fichiers- bibliothèques, vous trouverez peut-être que le deuxième lecteur de disquettes est rarement ou même jamais utilisé. Il est donc logique d'utiliser la mémoire!

Fermer le WorkBench

Veuillez noter que CLOSE WORKBENCH ne fonctionne pas si vous avez ouvert une fenêtre du CLI. Si vous voulez démarrer AMOS et fermer le workbench, vous devez utiliser les instructions du CLI suivantes:

```
RUN>NIL: AMOS1.2
ENDCLI
```

CLOSE WORKBENCH *(Fermer le Workbench)*

CLOSE WORKBENCH

est une commande qui ferme l'écran du Workbench et économise environ 40K de mémoire ainsi disponible pour vos programmes! Exemple:

```
Print Chip Free,Fast Free
Close Workbench
Print Chip Free,Fast Free
```

CLOSE WORKBENCH peut être exécutée lorsque vous êtes en mode direct ou dans un de vos programmes Basic. Voici par exemple une ligne typique de programme:

```
if Fast Free=0 Then Close Workbench
```

Cette instruction cherche s'il existe une mémoire étendue et ferme le Workbench au cas où il n'y en ait pas.

Fermer le WorkBench

Veuillez noter que CLOSE WORKBENCH ne fonctionne pas si vous avez ouvert une fenêtre du CLI. Si vous voulez démarrer AMOS et fermer le workbench, vous devez utiliser les instructions du CLI suivantes:

```
RUN>NIL: AMOS1.2
ENDCLI
```

CLOSE EDITOR *(Fermer la fenêtre de l'éditeur)*

CLOSE EDITOR

C'est une commande qui ferme la fenêtre de l'éditeur lorsque vous lancez vos programmes et économise ainsi plus de 28k de mémoire. En outre, il ne se produit absolument AUCUN effet sur les listings de vos programmes!

Si vous ne disposez pas d'une mémoire suffisante pour rouvrir la fenêtre une fois le programme terminé, AMOS effacera simplement la page d'écran présente et vous renverra l'écran standard PAR DEFAUT. Vous pouvez maintenant retourner facilement à l'éditeur à l'aide de la touche ESCape comme de coutume. Voilà comment une instruction tout à fait insignifiante devient absolument fantastique!

Les accessoires internes

Nous allons maintenant explorer les techniques générales nécessaires à la création de vos propres programmes accessoires. Ceux-ci ne sont en fait qu'une forme spécialisée des multiples programmes que nous avons étudiés un peu plus tôt. Comme vous vous y attendiez, ils peuvent accepter toutes les instructions standard Basic.

Les accessoires sont affichés directement sur votre écran-programme courant. La musique, les animations des sprites ou des bobs sont automatiquement supprimées de l'écran.

Votre accessoire devrait donc vérifier les dimensions et le type de cet écran en utilisant les commandes SCREEN HEIGHT, SCREEN WIDTH et SCREEN COLOUR lors de la phase d'initialisation. S'il n'accepte pas l'écran actuel, il peut vous obliger à ouvrir un nouvel écran pour la fenêtre de l'accessoire ou à effacer tous les écrans existants à l'aide d'une instruction DEFAULT.

Tous les banques mémoire qu'utilise votre accessoire sont complètement indépendantes du programme principal. S'il s'avère nécessaire de changer les banques depuis le programme courant, vous pouvez appeler une commande spéciale BGRAB.

BGRAB *(Saisie des banques utilisées par le programme courant)*

BGRAB b

BGRAB "emprunte" une banque du programme courant et le copie dans la même banque dans votre accessoire. Si la banque de cet accessoire existe déjà, il sera complètement effacé. Lorsque l'accessoire passe la main à l'éditeur, la banque que

vous avez saisie est automatiquement renvoyé à votre programme principal avec toutes les modifications effectuées. b est le nombre de banques compris entre 1 et 16.

Notez que cette instruction peut seulement être utilisée lorsque vous êtes dans un accessoire. Si vous essayez de l'intégrer à un programme normal, vous obtiendrez un message d'erreur.

PRUN *(Lancement d'un programme depuis la mémoire)*

PRUN "nom"

Exécute un programme Basic qui a été préalablement installé dans la mémoire de l'Amiga. Vous pouvez utiliser cette commande soit lorsque vous êtes en mode direct ou lorsque vous êtes dans un programme! En effet, PRUN est très similaire à l'appel classique d'une procédure, sauf qu'elle entraîne l'arrêt momentané de l'animation des bobs, des sprites ou de la musique.

Notez qu'il est impossible d'appeler deux fois le même programme dans la même session. Une fois appelé, toute autre instruction qui suit est rejetée.

Lorsque le programme repasse la main à votre accessoire, il vous faut réinitialiser votre écran. Le risque d'altération de vos écrans accessoires par la nouvelle routine sera ainsi évité. Référez-vous au fichier **EXEMPLE 3.3** du dossier MANUEL.

=PRG FIRST$ *(Lecture du premier programme chargé en mémoire)*

p$=PRG FIRST$ ("*.*")

Cette commande retourne le nom du premier programme Basic installé dans la mémoire de votre Amiga. Elle est utilisée conjointement avec la commande PRG NEXT$ pour créer une liste complète de tous les programmes actuellement disponibles.

=PRG NEXT$ *(Retourne le nom du programme suivant installé en mémoire)*

p$=PRG NEXT$

PRG NEXT$ est utilisée après l'exécution d'une commande PRG FIRST$ pour afficher tous les programmes installés dans la mémoire de l'Amiga. Lorsque vous avez atteint la fin de la liste, une valeur "" sera renvoyée par cette fonction. Voici un exemple:

```
N$=Prg First$ ("*.*")
While N$<>""
   Print "Programme ",N$
   N$=Prg Next$
Wend
```

=PSEL$ *(Appel d'un sélecteur de programme)*

n$=PSEL$("filtre "[défaut$,titre1$,titre2$]

PSEL$ appelle un sélecteur de programme qui est identique à celui que vous avez utilisé dans les commandes *Run Autre, Editer Autre, Charger Autres* et *Effacer Autres.* Il peut être utilisé de la même façon pour sélectionner un programme.. Le nom de ce programme sera communiqué dans la chaîne *n$.* Si vous avez suspendu l'exécution du programme par le sélecteur, n$ sera positionné sur une chaîne vide "".

La commande *filtre* détermine le type de programmes qui sera répertorié par cette instruction. Les valeurs typiques sont:

"*."	ACC"Répertorie tous les accessoires en mémoire.
"*."	AMOS"Affiche seulement les programmes AMOS qui ont été installés.
"*.*"	Répertorie tous les programmes existants en mémoire.

Voir la commande DIR pour plus de précisions sur le système.

défaut$	Contient le nom d'un programme utilisé implicitement.
titre1$,titre$	Contient une à deux lignes de texte qui seront affichées en haut du sélecteur.

Référez-vous au fichier **EXEMPLE 3.4** dans le dossier MANUEL si vous souhaitez une démonstration de cette instruction.

L'accessoire AIDE

Lorsque vous appuyez sur la touche HELP depuis la fenêtre de l'éditeur, AMOS exécute automatiquement un accessoire avec le nom HELP.ACC si celui-ci est disponible. A la différence des accessoires ordinaires, il est affiché directement dans la fenêtre de l'éditeur. Un accès spécial est prévu à la position courante du mot que vous éditez. L'adresse de ce mot est placée dans un registre d'adresses et elle peut être lue en utilisant la fonction AREG.

Les touches de commande de l'éditeur

Finalement, voici une liste complète des différentes touches de commande et leurs effets.

Touches spéciales

ESCape	Passe en mode direct

Touches d'édition

Rappel arrière	Efface le caractère se trouvant juste à la gauche du curseur.
DELete	Efface le caractère se trouvant juste dessous le curseur.
RETOUR	Fixe la ligne courante. Si vous changez de ligne et appuyez sur RETOUR, la ligne sera divisée en deux

	(cela se produit seulement si vous n'avez rien changé).
Shift+Retour arrière ou Ctrl+	YEfface la ligne courante et fait remonter le reste du texte.
Ctrl+U	Défaire. Restaure la dernière ligne lorsque vous êtes en mode *Superposition*.
Ctrl+Q	Efface le reste des caractères de la ligne où se trouve le curseur.
Ctrl+I	Insère une ligne à la position courante.

Les flèches de déplacement du curseur

Gauche	Avance le curseur d'un espace vers la gauche.
Droite	Recule le curseur d'un espace vers la droite.
Ascendante	Bouge le curseur d'une ligne vers le haut. Aucun effet est produit si le curseur se trouve sur la première ligne de l'écran.
Descendante	Bouge le curseur d'une ligne vers le bas.
Shift+flèche gauche	Positionne le curseur sur le mot précédent.
Shift+flèche droite	Positionne le curseur sur le mot suivant.
Shift+flèche ascendante	Déplace le curseur sur la première ligne de la page courante.
Shift+flèche descendante	Déplace le curseur sur la dernière ligne de la page courante.
Ctrl+flèche ascendante	Affiche la page précédente du texte à l'écran.
Ctrl+flèche descendante	Affiche la page suivante du texte à l'écran.
Ctrl+shift+flèche ascendante	Déplace le curseur au commencement du texte.
Ctrl+shift+flèche descendante	Déplace le curseur à la fin du texte.
Amiga+flèche ascendante	Fait remonter le texte sans bouger le curseur.
Amiga+flèche descendante	Fait descendre le texte sous le curseur.
Amiga+flèche gauche	Fait défiler le programme vers la gauche. Le curseur reste fixé sur la ligne courante.
Amiga+flèche droite	Déplace le texte vers la droite.

Commande du Programme

Amiga+S	Sauvegarde votre programme sous un nouveau nom.
Amiga+shift+S	Sauvegarde le programme sous le nom courant.
Amiga+L	Charge un programme.
Amiga+P	Pousse le programme courant en mémoire et crée un nouveau programme.
Amiga+F	Permet de basculer entre deux programmes sauvegardés en mémoire.
Amiga+T	Affiche le nouveau programme en mémoire. En répétant cette option, vous affichez touts les programmes actuellement en mémoire.

Couper-Coller

Ctrl+B	Marque le début d'un bloc.
Ctrl+E	Marque la fin d'un bloc.
Ctrl+C	Découpe le bloc. Charge le bloc en mémoire et l'efface de sa position courante. (Cette combinaison est également valable en mode direct, où elle S'ARRETE au programme courant.)
Ctrl+M	Déplace le bloc.
Ctrl+S	Sauvegarde le bloc en mémoire sans l'effacer en premier lieu.
Ctrl+P	Colle le bloc à la position courante du curseur.
Ctrl+H	Cache le bloc. L'effet de surbrillance disparaîtra du bloc choisi.

Les bornes

Ctrl+Shift+n	Définit un marqueur à la position courante du curseur. *n* doit être un chiffre du pavé numérique compris entre 0 et 9.
Ctrl+n	Se rend à un marqueur. Fait un saut à un marqueur préalablement posé. *n* doit être un chiffre du pavé numérique.

Chercher et Remplacer

Alt+flèche ascendante	Parcourt tout votre programme dans le sens arrière à la recherche de la ligne contenant la définition de la procédure ou du label.
Alt+flèche descendante	Parcourt tout votre programme dans le sens avant à la recherche de la ligne contenant la définition de la procédure ou du label.
Ctrl+F	Trouver. La commande vous demande d'entrer le mot que vous recherchez dans votre programme. Il affiche le prochain mot qu'il peut trouver depuis la position courante du curseur.
Ctrl+N	Rechercher. Utilisez cette fonction pour afficher la prochaine occurence de votre chaîne.
Ctrl+R	Remplacer. Si vous utilisez cette fonction avant celle de *Trouver*, vous serez invité à entrer une chaîne de remplacement. Toutefois, si vous avez démarré votre recherche avec *Trouver*, Ctrl+R remplacera votre mot par la chaîne de remplacement et cherchera la prochaine occurence.

Tabulations

Tab	Déplace toute la ligne à la position courante du curseur jusqu'au nouveau taquet de tabulation.

Shift+Tab	Déplace la ligne au taquet de tabulation suivant.
Ctrl+Tab	Pose les taquets de tabulation.

4 Principes Basic

Ce chapitre examine les règles de base utilisées pour élaborer les programmes AMOS Basic et vous montrer comment améliorer votre style de programmation à l'aide des procédures AMOS Basic.

Les variables

Les variables sont les noms utilisés pour désigner les emplacements de la mémoire dans un ordinateur. Ces emplacements contiennent les résultats des calculs effectués dans un de vos programmes.

C'est à vous de composer les noms de variables que vous voulez, ces derniers pouvant comprendre n'importe quelle chaîne de chiffres ou de lettres. Il existe seulement deux ou trois restrictions. Tous les noms de variables **doivent** commencer par une lettre et ne peuvent pas débuter par une instruction existant en AMOS Basic. Toutefois, vous êtes autorisé à utiliser ces mots clés à l'intérieur d'un nom. Ainsi, des variables telles que VPRINT ou SCORE conviennent.

Les noms de variables doivent être continus et ne peuvent pas être séparés par des espaces. Il est toutefois possible de remplacer un espace par un caractère "_" (signe moins).

Les noms suivants sont des noms admis:

AWHILE$,HIGH_SCORE,TEST_FLAG,HEIGHT#

La longueur maximale de ces noms de variable est de 255 caractères. Par exemple:

UN_NOM_TRES_LONG=10 : Rem Ceci est ok

Voici quelques exemples de noms interdits. Les parties interdites sont soulignés pour clarifier les choses.

WHILE$,5C,MODERN#,TOAD

Types de variables

AMOS Basic vous permet d'utiliser trois différents types de variables dans vos programmes.

Les nombres entiers

A la différence de la plupart des autres langages Basic, AMOS suppose tout d'abord que toutes les variables sont des nombres entiers. Par exemple, 1, 3 ou 8 sont des nombres entiers et conviennent parfaitement pour représenter les valeurs utilisées dans vos jeux.

Puisque les calculs en nombres entiers sont bien plus rapides que ceux en virgule flottante, l'utilisation de nombres entiers dans vos programmes peut améliorer grandement la vitesse d'exécution. Chaque nombre entier est mémorisé dans quatre

octets et peut être compris entre -147.483.648 et +147.483.648. Voici des exemples de variables de nombres entiers:

```
A,NUMBER,SCORE,LIVES
```

Les nombres réels

En AMOS Basic, ces variables sont toujours suivies d'un signe #. Les nombres réels peuvent contenir des valeurs fractionnelles telles que 3,1 ou 1,5. Celles-ci correspondent aux variables standard utilisées dans la plupart des autres versions Basic. Chaque variable réelle est mémorisée dans quatre octets et peut être comprise entre 1E-14 et 1E- 15. Toutes les valeurs sont précises à la septième décimale près. Voici un exemple:

```
P#, NUMBER#, TEST#
```

Les variables chaîne

Les variables chaîne contiennent des lettres et signes plutôt que des nombres. Ils se distinguent des variables normales par le signe $ ajouté à la fin du nom. La longueur de votre texte peut être compris entre 0 et 65.500 caractères. Voici un exemple de variable chaîne:

```
NAME$,PATH$,ALIEN$
```

Attribuer une valeur à une variable

Attribuer une valeur à une variable est facile. Il vous suffit de choisir un nom approprié et de lui affecter une valeur en utilisant le signe =.

```
VAR=10
```

La variable VAR reçoit ici une valeur de 10. En fonction de son type, la variable peut contenir un nombre ou un bloc de lettres et de signes. Pour affecter une chaîne à une variable, vous la délimitez par des guillemets comme suit:

```
A$="Bonjour"
```

Remarquez le signe $ après la lettre A. Il signale à AMOS que la variable contient des lettres et des signes et non un nombre.

Les tableaux

Toute liste de variables peut être regroupée sous la forme d'un tableau. Les tableaux sont produits en utilisant l'instruction DIM.

DIM *(Dimensionner un tableau)*

DIM var(x,y,z,...)

DIM définit un tableau de variables dans votre programme AMOS Basic. Ces tableaux peuvent avoir autant de dimensions que vous voulez, mais chaque dimension est limitée à un maximum de 65.000 éléments. Par exemple:

```
Dim A$(10),B(10,10),C#(10,10,10)
```

Pour accéder à un élément du tableau, il vous suffit de taper le nom du tableau suivi des nombres de l'index. Ces nombres sont séparés par des virgules et sont délimités par des parenthèses(). Notez que le premier nombre élément de ces tableaux est toujours zéro. Par exemple:

```
Dim ARRAY(10)
ARRAY(0)=10:ARRAY(1)=15
Print ARRAY(1);ARRAY(0)
15 10
```

Les constantes

Les constantes sont simplement des nombres ou des chaînes affectées à une variable ou utilisées dans un de vos calculs. Elles sont appelées constantes parce qu'elles ne changent pas pendant le cours de votre programme. Les valeurs suivantes sont toutes constantes:

```
1,42,3,141,"Bonjour"
```

Par défaut, toutes les constantes numériques sont considérées comme des nombres entiers. Toute affectation d'un nombre à virgule flottante à une variable entière provoquera automatiquement sa conversion en un nombre entier avant d'intégrer la variable. Par exemple:

```
A=3.141:PrintA
3
Print 19/2
9
```

Les constantes peuvent être également entrées en mémoire en utilisant une notation binaire ou hexadécimale. Les nombres binaires sont précédés du signe % et les nombres hexadécimaux du signe $. Voici diverses expressions du nombre 255.

Décimal:	**255**
Hexadécimal:	**$FF**
Binaire:	**%11111111**

Notez que tous les nombres que vous tapez en AMOS Basic sont automatiquement convertis en un format interne spécial. Lorsque vous listez vos programmes, ces nombres reprennent leur forme originale. Puisque AMOS Basic affiche tous les nombres de façon standard, de petits écarts peuvent souvent apparaître entre le nombre que

vous venez d'entrer et celui qui se trouve dans votre programme. Toutefois, la valeur du nombre restera exactement la même.

Les constantes en virgule flottante se distinguent des nombres entiers par une point décimale. Si ce point n'est pas utilisé, le nombre sera toujours considéré comme un nombre entier, même si ce nombre se présente au sein d'une expression en virgule flottante. Prenez l'exemple suivant:

```
For X=1 To 10000
   A#=A#+2
Next X
```

Chaque fois que l'expression de ce programme est évaluée, le "2" sera laborieusement converti en un nombre réel. Ainsi, de par sa structure, cette routine sera plus lente que le programme équivalent ci-dessous:

```
For X=1 To 10000
   A#=A#+2.0
Next X
```

L'exécution de ce programme est 25% plus rapide que celle du programme d'origine parce que la constante est maintenant mémorisée directement sous la forme d'une constante en virgule flottante. Il faut donc toujours se rappeler de placer un point décimale après une constante en virgule flottante même s'il s'agit d'un nombre entier. Si, par erreur, vous mélangez des nombres en virgule flottante et des nombres entiers, le résultat renvoyé sera toujours un nombre réel. Par exemple:

```
Print 19.0/2
9.5
Print 3.141+10
13.141
```

Les opérations arithmétiques

Les opérations arithmétiques suivantes peuvent être utilisées dans une expression numérique.

^	Elévation à la puissance
/et*	Diviser et multiplier
MOD	Opérateur modulo (donne le reste d'une division)
+ et -	Addition et soustraction
AND	ET logique
OR	OU logique
XOR	OU exclusif logique

Nous avons établi la liste de ces opérations par ordre de priorité décroissant. Cette priorité se rapporte à la suite dans laquelle les diverses unités d'une expression arithmétique sont évaluées. En règle générale, les opérations de priorité la plus élevée

seront calculées en premier. Voici un exemple pratique de la façon dont une expression est calculée:

```
Print 10+2*5-8/4+5^2
```

Cette instruction évalue les opérations dans l'ordre suivant:

5^2 (égal à 5*5)	= 25
2*5	= 10
8/4	= 2
10+10	= 20
20-2	= 18
18+25	= 43

Pour obtenir une différente évaluation, il vous faut simplement délimiter, à l'aide de parenthèses, les parties de l'expression que vous souhaitez exécuter en premier:

```
Print(10+2)*(5-8/4+5)^2
```

Cette instruction vous donne le résultat 12*(8^2) ou 12*64 ou 768. Comme vous pouvez le constater, la simple intégration de deux couples de parenthèses a complètement changé le sens de l'expression.

Puisque nous étudions les opérations de calcul, il serait bon de mentionner trois instructions simples qui peuvent accélérer considérablement vos programmes.

INC *(Ajouter 1 à une variable entière)*

INC var

INC incrémente de 1 une variable entière en utilisant une seule instruction microprocesseur. Elle est logiquement équivalente à l'expression var=var+1, mais elle est bien plus rapide. Exemple:

```
A=10:Inc A:Print A
11
```

DEC *(Soustraire 1 à une variable entière)*

DEC var

Cette instruction soustrait 1 à la variable entière *var*. Exemple:

```
A=2
Dec A
Print A
1
```

ADD. *(Addition rapide de nombres entiers)*

ADD v,exp [, base TO top]

Cette instruction de forme classique ajoute immédiatement le résultat de l'expression *exp* à la variable nombre entier *v*. Elle est équivalente à la ligne: V=V+EXP

La seule importante différence entre les deux libellés est que l'exécution de ADD est environ 40% plus rapide. Notez que la variable *v* doit être un nombre entier. Exemple:

```
Timer=0
For X=1 To 1000
   Add T,X
Next X
Print T,Timer
500500 7
```

La deuxième variante de ADD est un petit peu plus compliquée. Elle est en fait identique au code suivant:

```
V=V+A
If V<Base Then V=TOP
If V>Top Then V=BASE
```

Comme la première version de ADD, cette commande est bien plus rapide que les instructions indépendantes. Voici un exemple:

```
Dim A(10)
For X=0 To 10:A(X)=X:Next X
V=0
Repeat
   Add V,1,1 To 10
Print A(V)
Until V=100:rem C'est une boucle infinie puisque V est toujours
inférieure à 10!
```

Vous pouvez donc constater que la commande ADD est idéale pour la manipulation de boucles circulaires ou répétitives dans vos jeux.

Opérations en chaînes

Comme la plupart des versions du Basic, AMOS vous permet heureusement d'ajouter deux chaînes à la fois.

```
A$="AMOS"+"Basic"
Print A$
AMOS Basic
```

Mais AMOS vous permet aussi bien d'effectuer une soustraction. Cette opération s'effectue en supprimant toutes les occurences de la deuxième chaîne depuis la première. Exemples:

```
Print "AMOS BASIC"-"S"
AMO BAIC
Print "AMOS BASIC"-"AMOS"
BASIC
Print "Une chaîne de caractères"-" "
UneChaînedeCaractères
```

Des comparaisons entre les deux chaînes sont établies caractère par caractère en utilisant la valeur Ascii des lettres appropriées. Exemples:

```
"AA"<"BB"
"Nomfichier"="Nomfichier"
"X&"> "X#"
"BONJOUR" < "bonjour"
```

Tapez le programme suivant:

```
Input "Ecrivez votre prénom";C$
Input "Ecrivez votre nom";S$
If C$>S$ Then Print S$;" ";C$ Else Print C$;" ";S$
```

Les paramètres

Les valeurs que vous entrez dans une instruction AMOS Basic sont connues sous le nom de *paramètres,* c'est-à-dire:

```
Inc N
Add A,10
Ink 1,2,3
```

Les paramètres contenus dans les instructions ci-dessus sont respectivement N, A, 10, 1, 2 et 3. Quelques-uns des paramètres d'une commande peuvent être quelques fois omis dans une instruction. Dans ce cas, toute valeur inutilisée recevra automatiquement un nombre par défaut. Prenez l'exemple suivant:

```
Ink 5,,
```

Cette instruction change la couleur de l'encre sans affecter la couleur du papier ou la couleur de tracé. Remarquez la place normale des virgules, même si les valeurs elles-mêmes ont été omises. AMOS utilise ces virgules pour déterminer l'ordre d'entrée des paramètres dans l'instruction, ce qui vous permet d'introduire une valeur au milieu d'une commande comme par exemple:

```
Ink,3,
```

L'instruction Ink sélectionne maintenant la couleur du papier en ne modifiant pas la couleur de l'encre ou de tracé.

Le même principe peut être également appliqué à de nombreuses autres instructions AMOS Basic. A condition que vous vous rappeliez de placer les virgules à leur position d'origine, vous pouvez utiliser cette technique pour vous éviter un surcroît de travail de frappe inutile dans vos programmes. Si un paramètre est d'importance capitale, un appel *interdit* de *fonction* se *présentera* à vous. Ainsi cela vaut bien la peine de mettre en pratique les diverses combinaisons.

Les numéros de ligne et les étiquettes

Dans les premières versions du Basic, chaque programme *devait* commencer par un *numéro*. Ce numéro de ligne faisait office de déstination pour les instructions GOTO ou GOSUB. Il était également utilisé par l'éditeur Basic. Bien que cette approche soit correcte (elle était même utilisée en STOS Basic), elle n'est pas vraiment nécessaire avec AMOS. En AMOS Basic, les numéros de ligne sont tout à fait optionnels; ils sont seulement prévus à des fins de compatibilité avec le STOS Basic.

Il se peut que vous vous demandiez comment vous pouvez utiliser GOTO ou GOSUB sans les numéros de ligne. Eh bien, vous pouvez les remplacer en utilisant des *étiquettes*.

Les étiquettes

Les étiquettes sont utilisées pour marquer un point dans vos programmes AMOS Basic. Elles se composent d'une chaîne de caractères formée suivant les règles des variables AMOS. Les étiquettes doivent toujours être placées au début d'une ligne, et doivent être suivies du signe : (deux points). Il ne doit pas y avoir d'espaces entre l'étiquette et les deux points. Sinon l'étiquette sera considérée comme une procédure et vous obtiendrez un message d'erreur *procédure indéfinie*. Voici un exemple simple:

```
TESTLABEL: Rem C'est une étiquette
Print "Salut"
Goto TESTLABEL
```

Ce programme imprime le mot "Salut" de façon répétée à l'écran. Le programme peut être arrêté en appuyant sur Control+C.

Les étiquettes sont bien plus faciles à lire que les numéros de ligne. Nous vous conseillons donc de les utiliser fréquemment dans vos programmes AMOS Basic.

Les procédures

Si vous avez déjà essayé d'écrire un très long programme en Basic, vous vous êtes sans doute rendu compte combien il peut être facile de se perdre complètement à mi-chemin. De nos jours, la plupart des programmeurs professionnels divisent leurs programmes en de petits modules connus sous le nom de procédures.

Les procédures vous permettent de concentrer vos efforts sur un problème à la fois sans que votre esprit soit tourmenté par le reste de votre programme. Une fois que vous avez écrit vos procédures, vous pouvez rapidement les intégrer au programme que vous venez d'élaborer.

Les programmes intégrant des procédures sont faciles à écrire, faciles à changer et faciles à mettre au point. Les procédures AMOS Basic sont des modules totalement indépendants du programme et peuvent avoir leurs propres lignes de programme, leurs propres variables et même leurs propres libellés de données. Ainsi, vous n'avez absolument aucune excuse pour ne pas en faire grand usage dans vos programmes AMOS Basic.

PROCEDURE *(Créer une procédure AMOS Basic)*

Procedure NAME [Liste de paramètres]
: :
End Proc[Expression]

Cette instruction définit une procédure AMOS Basic appelée *NAME. NAME* est une chaîne de caractères qui identifie la procédure. Elle est construite exactement de la même façon qu'une variable Basic normale. Notez qu'il est parfaitement convenable d'utiliser les mêmes noms pour les variables de procédures et les étiquettes. AMOS déterminera automatiquement l'objet auquel vous vous rapportez à partir du contexte de la ligne.

Les procédures sont similaires aux commandes GOSUB que vous avez rencontrées dans les premières versions du Basic. Voici un exemple de procédure AMOS simple.

```
Procedure ANSWER
  Print "Quarante-deux!"
End Proc
```

Constatez que la procédure a été terminée avec un ordre *END PROC*. Vous devriez également notez que la Procédure et les ordres End Proc sont tous les deux placés sur leurs lignes respectives. Ceci est obligatoire.

Si vous tapez la procédure précédente en AMOS Basic comme elle se présente et faites fonctionner le programme, rien ne se produira: vous n'avez pas appelé la nouvelle procédure de votre programme Basic. Pour l'accomplir, il vous suffit donc d'entrer son nom à l'endroit approprié de votre programme. A titre d'exemple, entrez la ligne suivante au début du programme et faites fonctionner ce dernier pour trouver le résultat de la procédure.

```
ANSWER
```

Important! Lorsque vous utilisez plusieurs procédures sur la même ligne, il est conseillé de prévoir un espace supplémentaire à la fin de chaque instruction. Vous évitez ainsi que la procédure soit confondue avec une étiquette. Par exemple:

```
TEST : TEST : TEST:Rem effectue trois fois la procédure test
TEST:TEST:TEST: Rem définit l'étiquette TEST et exécute TEST deux fois
```

Vous pouvez également faire précéder vos appels de Procédure de l'ordre Proc suivant:

```
Proc ANSWER
```

Par exemple:

```
Rem Vous montre certainement qu'une Procédure est appelée,
Rem pas simplement une commande.
Proc ANSWER
Rem Vous pouvez obtenir le même libellé sans le Proc
ANSWER
Procedure ANSWER
   Print "Quarante-Deux!"
End Proc
```

Si vous lancez à nouveau ce programme, la procédure sera exécutée et la réponse sera affichée à l'écran. Bien que la définition de la procédure soit positionnée à la fin du programme, vous pouvez la placer à l'endroit que vous désirez. Si AMOS rencontre un ordre de Procédure, il installe la procédure et se branche immédiatement au End Proc finale. Cela signifie que vous ne courrez pas le risque d'exécuter votre procédure par erreur. Lorsque vous avez créé une procédure et que ses essais vous satisfont, vous pouvez la supprimer du listing en utilisant l'option plier de votre menu principal.

Le fait de plier *réduit* l'apparente complexité de vos programmes et vous permet de développer de longs programmes sans avoir à vous occuper de détails insignifiants. Vous pouvez restaurer le listing des procédures à l'écran à tout moment en sélectionnant l'option déplier du menu.

Les variables locales et globales

Toutes les variables que vous définissez à l'intérieur de vos procédures sont indépendantes des autres variables utilisées dans votre programme. Ces variables sont dites locales en regard de votre procédure particulière. Voici un exemple illustrant ceci:

```
A=1000:B=42
TEST
Print A,B
Procedure TEST
   Print A,B
End Proc
```

Il devrait être évident que les noms A et B se rapportent à des variables complètement différentes selon qu'elles ont été utilisées en dedans ou en dehors de la procédure TEST. Les variables qui apparaissent en dedans ou en dehors d'une procédure sont globales et sont accessibles seulement lorsque vous vous trouvez dans la procédure. Prenons un autre exemple:

```
Dim A(100)
For V=1 to 100: A(V)=V:Next V
TEST_FLAG=1
APRINT
```

```
End
Procedure APRINT
   If TEST_FLAG=1
      For P=1 To 100
         Print A(P)
      Next P
   Endif
Endproc
```

Ce programme peut sembler sans malice et pourtant il renferme deux erreurs fatales.

Tout d'abord, la valeur TEST_FLAG de la procédure recevra toujours la valeur zéro. Ainsi la boucle se trouvant entre IF et ENDIF ne s'effectuera jamais parce que le TEST_FLAG utilisé dans la procédure est complètement indépendant de celui défini dans le programme principal. Comme toutes les variables, la valeur zéro lui est automatiquement affectée lors de sa première utilisation.

En outre, le programme ne fonctionnera pas! Puisque le tableau général A() a été défini en dehors de APRINT, AMOS Basic signalera immédiatement un message *d'erreur de tableau non dimensionné* à la ligne:

```
Print A(P)
```

Ce type d'erreur est vraiment très facile à commettre. Aussi, il est important de considérer les procédures comme des programmes indépendants dotés de leur propre jeu indépendant de variables et d'instructions. Ne tombez pas dans le piège en utilisant les mêmes noms de variables en dedans et en dehors d'une procédure, sinon vous pourriez être amené à croire que ce sont les mêmes variables et vous seriez conduit à commettre des erreurs inexplicables dans vos programmes.

Heureusement, il existe deux ou trois extensions de ce système facilitant le transfert des informations entre une procédure et votre programme principal. Une fois que vous vous êtes familiarisé avec ces commandes, vous n'aurez guère de problèmes à utiliser correctement les procédures dans vos programmes.

Le paramètres et les procédures

Une des possibilités est d'intégrer une liste de "définitions de paramètres" dans votre procédure. Un groupe de variables locales pouvant être chargé directement à partir du programme principal est ainsi formé. Voici un exemple:

```
Procedure BONJOUR[NOM$]
PRINT "Bonjour";NOM$
End Proc
```

La valeur devant être chargée dans NOM$ est entrée entre crochets et fait partie de l'appel de la procédure. Ainsi, la procédure BONJOUR pourrait s'exécuter de la façon suivante:

```
Rem Charge N$ dans NOM$ et entre la procédure
Input "Quel est votre nom";n$
```

```
BONJOUR[N$]
Rem Charge la chaîne littérale "Stephen" dans NOM$ et appelle
BONJOUR
BONJOUR["Stephen"]
```

Vous pouvez donc constater que le système de paramètrage est polyvalent et il est valable tout aussi bien avec des variables qu'avec des constantes. Seul le type de variables est important.

Ce processus peut être utilisé pour transférer des variables nombres entiers, des variables réelles ou des variables chaînes. Toutefois, vous ne pouvez pas produire des tableaux entiers avec cette fonction. Si vous voulez entrer plusieurs paramètres, il vous faudra séparer vos variables par des virgules. Par exemple:

```
Procedure POWER[A,B]
Procedure MERGE[A$,B$,C$]
```

Ces procédures peuvent être appelées en utilisant les lignes suivantes:

```
POWER[10,3]
MERGE["Un","Deux",Trois"]
```

Les variables partagées

Une autre façon de faire circuler les données entre une procédure et le programme principal est d'utiliser l'instruction SHARED.

SHARED *(Définir un bloc de variables globales)*

SHARED liste de variables

SHARED est placé dans une procédure et saisit un bloc de variables AMOS Basic séparées par des virgules. Ces variables sont maintenant considérées comme des variables globales et sont directement accessibles depuis le programme principal. Tout tableau que vous déclarez de cette façon devrait évidemment avoir été dimensionné dans votre programme principal. Exemple:

```
A=1000:b=42
TEST
Print A,B
Procedure Test
   Shared A,B
   A=A+B:B=B+10
End Proc
```

TEST peut maintenant lire des informations des variables globales A et B ou en écrire. Si vous voulez partager un tableau, vous devriez le définir ainsi:

```
hared A(),B#(),C$():rem Partage tableaux A,B# and C$
```

GLOBAL *(Déclarer une liste de variables globales depuis le programme principal)*

GLOBAL liste de variables

Lors de la rédaction d'un long programme, il est courant d'avoir un certain nombre de procédures partageant le même jeu de variables globales. Ceci permet d'avoir une méthode simple de transfert de grandes quantités d'informations entre les différentes procédures. Afin de simplifier ce processus, nous avons prévu une simple commande pouvant être utilisée directement dans votre programme principal. GLOBAL définit une liste de variables qui sont accessibles dans tout le corps de votre programme Basic, sans avoir recours à l'ordre explicite SHARED dans les procédures. Exemple:

```
A=1000 : B=42
Global A,B
TEST1
Print A,B
TEST2
Print A,B
Procedure TEST1
   A=A+B : B=B+10
End Proc
Procedure TEST2
   A=A*B : B=B+10
End Proc
```

Renvoyer des valeurs depuis une procédure

Si une procédure doit renvoyer une valeur qui lui est seulement locale, elle doit utiliser la commande suivante afin de pouvoir renseigner la commande appelante PROCEDURE sur la position de la variable locale.

PARAM *(Renvoyer un paramètre depuis une procédure)*

PARAM

Les fonctions PARAM vous permettent de renvoyer simplement un résultat d'une procédure. Elles saisissent le résultat d'une expression optionnelle dans l'instruction END PROC, et le renvoient dans l'une des variables PARAM,PARAM# ou PARAM$ en fonction de son type. Exemple:

```
MERGE_CHAINES["Amos","","Basic"]
Print PARAM$
Procedure MERGE_CHAINES[A$,B$,C$]
   Print A$,B$,C$
End Proc[A$+B$+C$]
```

Notez que END PROC peut seulement renvoyer un seul paramètre de cette façon. Les fonctions PARAM contiendront toujours le résultat de la procédure qui a été exécutée en dernier. Voici un autre exemple illustrant cette fois-ci l'utilisation de la fonction PARAM#.

```
CUBE[3.0]
Print Param#
Procedure CUBE[A#]
   C#=A#*A#*A#
Endproc[C#]
```

Quitter une procédure

POP PROC *(Quitter immédiatement une procédure)*

POP PROC

D'une façon générale, les procédures reviendront seulement au programme principal si elles sont parvenues à l'instruction END PROC. Toutefois, il se peut que vous ayez besoin de quitter une procédure d'urgence. Dans ce cas, vous pouvez utiliser la fonction POP PROC pour sortir immédiatement. Exemple:

```
Procedure TERMINATE
   For RASANT=1 TO 100000
      If RASANT=10 Then Pop Proc
   Next RASANT
   Print "Cette ligne n'est jamais exécutée"
End Proc
```

Ordres de DONNEES locales

Tout "data" défini dans une de vos procédures est complètement indépendant de ceux du programme principal. Ceci signifie que chaque procédure peut avoir ses propres zones de données. Exemple:

```
Read A$:Print A$,
EXAMPLE
Read B$:Print B$
Data "AMOS","étonnant!!"
Procedure EXAMPLE
   Read X$,Y$
   Print X$,Y$
   Data "Basic", "est"
End Proc
```

Conseils et suggestions

Voici quelques lignes directrices qui vous aideront à tirer profit de vos procédures AMOS Basic:

Une procédure est parfaitement autorisée à s'appeler, mais cette récurrence est limitée par l'encombrement des variables locales. Si votre programme est à court de mémoire, vous obtiendrez le message d'erreur approprié.

- Toutes les variables locales sont automatiquement **supprimées** à la fin de l'exécution de la procédure.

```
Procedure ADD
        A=A+1:Print A
End Proc
```

Peu importe le nombre d'appels de cette procédure, celle-ci affichera toujours la même valeur (1).

- Les procédures AMOS sont équivalentes aux sous-programmes créées en utilisant les commandes SUB d'Amiga Basic. La seule importante différence est que vous ne pouvez pas transférer des tableaux en tant que paramètres. Si vous devez accéder à un tableau lorsque vous vous trouvez dans une procédure, il vous faut le déclarer en tant que *shared*.

Les banques mémoire

AMOS Basic comprend un certain nombre de puissantes fonctions pour manipuler les sprites, les bobs et la musique. Les données mises en jeu par ces fonctions doivent être mémorisées avec le programme Basic. AMOS Basic utilise à cette fin un jeu spécial de 15 blocs de mémoire appelés *banques*.

Chaque banque est désignée par un nombre unique compris entre 1 et 15. La plupart de ces banques peuvent être utilisées pour tous les types de données, mais certaines sont seulement réservées pour un type d'informations telles que les définitions de sprites. Toutes les images de sprites sont mémorisées dans la banque 1. Elles peuvent être chargées en mémoire en utilisant par exemple la ligne suivante:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
```

Il existe deux formes différentes de banque mémoire: la banque mémoire *permanente* et la banque mémoire *temporaire*. Les banques permanentes doivent être définies une seule fois et elles sont en conséquence sauvegardées automatiquement avec votre programme. Les banques temporaires ne sont pas aussi rémanentes et sont réinitialisées chaque fois qu'un programme est lancé. En outre, à la différence des banques permanentes, les banques temporaires peuvent être effacées de la RUN.

Les types de banques mémoire

AMOS Basic supporte les types suivants de banque mémoire:

Class	Mémorise	Restrictions	Type
Sprites	Sprite ou définitions de bob	Banque 1 seulement	Permanente
Icônes	Contient les définitions d'icônes	Banque 2 seulement	Permanente
Musique	Contient les données de la piste sonore	Banque 3 seulement	Permanente
Amal	Utilisée pour les données AMAL	Banque 4 seulement	Permanente
Echantillons	Les données d'échantillons	Banques 1-15	Permanente
Menu	Mémorise la définition du menu	Banques 1-15	Permanente
Chip Work	Espace de travail temporaire	Banquess 1-15	Temporaire
Chip Data	Espace de travail permanent	Banques 1-15	Permanente
Work rapide	Espace de travail temporaire	Banques 1-15	Temporaire
Data rapide	Espace de travail permanent	Banques 1-15	Permanente

RESERVE *(Réserver une banque)*

RESERVE AS type,bank,length

Les banques utilisées par vos sprites ou bobs sont automatiquement affectées par AMOS. La commande RESERVE vous permet de créer n'importe quelle autre banque mémoire dont vous pourriez avoir besoin. Chaque type de banque a sa propre version de l'instruction RESERVE.

RESERVE AS WORK nobanque,long

Cette commande réserve la *longueur* des octets dans un espace de travail temporaire. Si possible, cette zone mémoire sera réservée en utilisant une mémoire rapide; vous ne devez donc pas appeler cette commande conjointement avec les instructions qui sont nécessaires pour accéder à la puce de manipulation d'objets d'Amiga.

RESERVE AS CHIP WORK nobanque,long

Cette commande affecte un espace de travail de taille *longueur* en utilisant la CHIP-RAM. Vous pouvez vérifier s'il y a assez de CHIP-RAM disponible à l'aide de la fonction CHIP FREE.

RESERVE AS DATA nobanque,long

Cette commande met de côté une banque mémoire permanente longueur codée en octets. Cette zone de données sera réservée en utilisant la mémoire rapide si elle est disponible.

RESERVE AS CHIP DATA nobanque,long

Cette commande réserve les octets *longueur* de la mémoire dans la CHIP RAM. Cette banque sera automatiquement sauvegardée avec vos programmes AMOS.

Le *banque* peut être attribuée un numéro compris entre 1 et 15. Puisque les banques sont normalement réservées par le système, il est plus sage de les laisser seules. Notez que la seule limite imposée à la longueur d'une banque est la quantité de mémoire disponible.

LISTBANK *(Lister les banques en cours d'utilisation)*

LISTBANK liste les nombres de banques actuellement réservées par un programme, ainsi que leur position et leur taille. Le listage se présente selon le format suivant:

Nombre	Type	Départ	Longueur
1	- Sprites	S: $040F60	L:$00002F
2	- Travail	S: $05F7A0	L:$014000

S: = L'adresse de départ de la banque en hexadécimal

L: = La longueur de la banque en hexadécimal.

Normalement, la longueur d'une banque est renvoyée en octets, mais dans le cas de sprites et d'icônes, la valeur représente le nombre total d'images de la banque. Cela s'explique par le fait que la mémorisation de chaque image peut s'effectuer dans n'importe quelle partie de la mémoire de votre Amiga; la banque n'est donc pas un bloc de mémoire continu. Ne sauvegardez donc pas une banque de sprites en utilisant la commande BSAVE mais en faisant un simple appel à SAVE "nomfichier.ABK".

Effacer des banques

Au cours d'un programme, il se peut que vous ayez besoin d'effacer certaines banques de la mémoire afin de pouvoir charger des données supplémentaires. Vous voulez peut-être changer les sprites pour une nouvelle partie du jeu ou modifier le morceau de musique qui doit être joué. La commande ERASE vous donne un contrôle rapide de la suppression des données.

ERASE *(Effacer ue banque)*

ERASE b

ERASE efface le contenu d'une banque de mémoire. La banque numéro b peut être comprise entre 1 et 15. Notez que toute mémoire utilisée par cette banque est en conséquence libérée et se trouve à la disposition de votre programme.

Les fonctions de paramètre de banque

En utilisant des commandes telles que poke, doke et loke, vous pouvez accéder directement aux données de la banque et trouver l'adresse et la taille d'une banque en mémoire.

=START *(Renvoyer l'adresse de départ d'un banque)*

s=START(b)

Cette fonction renvoie l'adresse de départ d'une banque numéro *b*. Une fois qu'elle a été réservée, son emplacement dans la mémoire ne changera donc jamais. Ainsi, le résultat de cette fonction demeurera fixe pendant toute l'existence de la banque. Exemple:

```
Reserve As Work 3,2000
Print Start(3)
```

=LENGTH *(Renvoyer la longueur d'une banque)*

l=LENGTH(b)

La fonction LENGTH renvoie la longueur d'une banque numéro *b* en octets. Si la banque contient des sprites, le nombre de sprites ou d'icônes sera alors renvoyé. Une valeur zéro indique que la banque *b* n'existe pas. Exemple:

```
Reserve as work 6,1000
Print Length(6)
Erase 6
Print Length(6)
```

Charger et sauvegarder les banques

Certains programmes demandent de nombreuses banques d'informations; les programmes de jeux d'aventure en sont un bon exemple. Dans un des jeux, les différents emplacements sont utilisés pour contenir les divers graphismes et sons. Un Amiga 500 aurait beaucoup de difficultés à conserver en même temps toutes ces données et il est donc préférable de charger les données au moment où vous souhaitez les utiliser.

LOAD *(Charger une ou plusieurs banques)*

LOAD "nomfichier"[,n]

L'effet de cette commande varie en fonction du type de fichier que vous chargez. Si le fichier mobilise plusieurs banques, toutes **les** banques mémoire courantes seront alors effacées avant que les nouvelles banques soient chargées depuis le disque. Toutefois, si vous chargez uniquement une banque, seule cette banque sera remplacée. Le

paramètre optionnel spécifie la banque devant recevoir vos données. S'il est omis, les données seront alors chargées dans la banque à partir de laquelle elles furent initialement sauvegardés.

Les banques de sprites sont considérées quelque peu différemment. Dans ce cas, le paramètre n permute deux modes indépendants de chargement. Si n est omis ou si *n* est affecté d'une valeur zéro, la banque courante sera entièrement remplacée par les nouveaux sprites. Toute autre valeur de *n* force les nouveaux sprites à être *annexés* à cette banque, ce qui vous permet de combiner plusieurs fichiers de Sprites dans le même programme. Exemple:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
```

SAVE *(Sauvegarder une ou plusieurs banques sur le disque)*

SAVE "nomfichier"[,n]

La commande SAVE sauvegarde vos banques mémoire sur le disque. Il existe deux variantes possibles:

```
SAVE "nomfichier.ABK"
```

Cette commande sauvegarde tous les banques couramment définies dans un seul fichier de votre disque.

```
SAVE "nomfichier.ABK",n
```

Celle-ci, de forme développée, sauvegarde la banque mémoire numéro *n*. Vous devez également vous assurer que le suffixe ABK figure après le nom du fichier car il vous permettra d'identifier le fichier contenant une ou plusieurs banques mémoire.

BSAVE *(Sauvegarder un bloc-mémoire non formaté sous format binaire)*

BSAVE fichier$, début TO fin

La mémoire conservée entre début et fin est sauvegardée sur le disque sous la forme de fichier$. Ces données sont sauvegardées sans formatage spécial. Exemple:

```
Bsave "Test",Start(7) To Start(7)+Length(7) : Rem Sauvegarde une banque mémoire
```

L'exemple ci-dessus sauvegarde les données dans la banque mémoire 7 du disque. La différence existant entre ce fichier et un fichier sauvegardé en tant que banque normale est que SAVE écrit un en-tête spécial contenant les informations concernant la banque. L'en-tête n'est pas écrit pour un fichier sauvegardé au moyen de BSAVE; il ne peut donc pas être chargé en utilisant LOAD.
Attention: Le stockage des banques de sprites et des banques d'icônes n'est pas localisé dans une seule tranche de mémoire. Chaque objet peut résider dans un

emplacement quelconque de la mémoire. Vous ne pouvez pas simplement sauvegarder la banque mémoire en utilisant BSAVE en raison de ce système souple de conservation de données.

BLOAD *(Charger les informations binaires dans une adresse spécifiée ou une banque)*

BLOAD fichier$, adr

La commande BLOAD charge un fichier binaire en mémoire. Elle ne modifie pas du tout les informations d'entrées. Il existe deux variantes de cette fonction.

Bload fichier$, adr

Fichier$ sera chargé à partir du disque dans l'adresse adr.

Bload fichier$, banque

Fichier$ sera chargé dans la *banque*. Cette banque doit avoir été préalablement réservée, sinon un message d'erreur sera produit. Soyez également sûr de ne pas charger un fichier qui soit plus grand que la banque réservée, autrement il dépassera la capacité de la banque et commencera à altérer les autres zones de la mémoire.

BANK SWAP *(Permuter deux banques en mémoire)*

BANK SWAP numéro1,numéro2

Cette instruction permute les pointeurs de deux banques. C'est utile si vous voulez transformer une banque d'icônes en une banque de sprites. Exemple:

Bank Swap 1,2

ou avoir plus d'une banque de musique à la fois, par exemple:

Bank Swap 3,5

Fragmentation de la mémoire

Après une longue session d'édition, il se peut que vous obteniez un message d'erreur "Mémoire pleine", même si la ligne d'information vous laisse supposer que vous disposez d'une grande quantité de mémoire disponible. Ce **n'est pas** une imperfection de l'AMOS Basic. C'est un effet inévitable du système d'allocation de la mémoire de l'Amiga.

Le système d'allocation de la mémoire de l'Amiga se présente un peu comme un gâteau. Une fois que vous avez coupé une tranche, vous ne pouvez absolument pas la replacer à sa position d'origine. Chaque fois que vous effectuez une réservation de mémoire, une seule tranche est coupée du gâteau-mémoire. Si vous renvoyez cette

mémoire au système, elle est placée sur une "pile" spéciale de toutes les tranches actuellement inutilisées.

La prochaine fois que vous réservez une zone de mémoire, l'Amiga cherchera la tranche de la bonne taille en vérifiant chacune d'entre elles. S'il trouve une section qui soit plus importante que celle que vous ayez demandée, il la coupera automatiquement en deux morceaux, gardant toute mémoire inutilisée pour lui-même. Au bout d'un moment, la mémoire se fragmentera en un grand nombre de très petites tranches. Si vous en redemandez, vous ne pourrez même pas en trouver dans une seule tranche et vous obtiendrez un message d'erreur plutôt gênant, "Mémoire pleine".

L'unique manière sûre de résoudre ce problème est d'éteindre votre Amiga et de le réinitialiser. Ceci remettra la zone mémoire à son état initial et vous pourrez attribuer la mémoire comme vous l'entendez.

Notez que la difficulté ci-dessus apparaît seulement pendant le calcul d'adresse. Si vous réservez toute votre mémoire au début de vos programmes, vous n'aurez jamais à vous soucier de ce problème dans aucun de vos programmes.

Trouver de la place pour vos variables

Toutes les variables sont conservées par défaut dans une zone mémoire d'une longueur exacte de 8k. Bien que celle-ci puisse sembler incroyablement petite, elle peut facilement contenir approximativement 2 pages de lettres et de signes ou 2000 nombres. Nous l'avons intentionnellement configurée la plus petite possible afin de maximiser la quantité d'espace disponible pour vos écrans et bancs mémoire.

Cependant, ne paniquez pas! La taille de cette zone peut être augmentée directement depuis vos programmes Basic en utilisant une simple commande SET BUFFER. La seule limite physique imposée à la taille de vos tableaux et aux variables en chaîne est la quantité de mémoire que vous avez installée dans votre ordinateur.

SET BUFFER *(Définir la taille de la zone de variables)*

SET BUFFER n

Cette commande fixe la taille de la zone de variables dans votre programme courant à *n* kilobytes. Ceci doit être la PREMIERE instruction de votre programme (à l'exception des Rems), sinon vous obtiendrez le message d'erreur approprié. Pour illustrer cette fonction, référez-vous au fichier **EXEMPLE 4.1** du dossier MANUEL.

La commande SET BUFFER doit être utilisée dans votre programme *lorsque* vous *obtenez* un message d'erreur *espace* de chaîne insuffisant. Augmentez la valeur par tranche de 5k jusqu'à ce que le message d'erreur disparaîsse. Si vous êtes à court de mémoire pendant ce processus, il vous faudra sans doute réduire d'une certaine façon les besoins de votre programme. Voir les commandes CLOSE WORKBENCH et CLOSE EDITOR pour plus de précisions.

=FREE *(Renvoyer la quantité de mémoire disponible dans la zone de variables)*

f=FREE

FREE renvoie le nombre d'octets étant actuellement disponibles pour vos variables.

Cette valeur peut être augmentée si besoin en utilisant la commande précédente SET BUFFER.

Chaque fois que FREE est appelée, la zone de variables est réorganisée pour donner le maximum d'espace à vos variables. Ce processus est connu sous le nom de *ramasse-miettes* et il est généralement effectué automatiquement.

En raison de la puissance de l'AMOS Basic, toute la procédure est pratiquement accomplie instantanément. Mais si votre zone de variables est très grande et si vous utilisez beaucoup de chaînes, il se pourrait que la routine ramasse-miettes prenne plusieurs secondes pour s'exécuter. En théorie, elle pourrait amener un retard inattendu dans l'exécution de vos programmes. Puisque la routine ramasse-miettes est vraiment très importante, (comme dans la réalité!) il se peut que vous ayez besoin d'ajouter un appel explicite de la commande FREE lorsqu'il est susceptible de causer le moins de dérangement dans votre programme.

5 Opérations sur les chaînes

AMOS Basic est livré avec une gamme complète d'instructions de manipulation de chaînes. Nous avons pris le soin d'utiliser la syntaxe classique du Basic. Ainsi, si vous êtes un programmeur confirmé en Basic, vous vous serez déjà familiarisé avec la plupart de ces commandes.

=LEFT$= *(Renvoyer les caractères d'extrême gauche d'une chaîne)*

d$=LEFT$(s$,n)
LEFT$(d$,n)=s$

LEFT$ lit les n premiers caractères se trouvant à la gauche de la chaîne *s$* et les copie dans la chaîne de destination *d$*.

Comme vous pouvez le constater, cette commande se présente sous deux formes générales. La première version est une fonction qui créé une nouvelle chaîne de destination *d$* à partir des *n* premiers caractères de la source *s$*. Exemples:

```
Print Left$("AMOS Basic",4)
AMOS
A$=Left$("0123456789ABCDEF",10)
Print A$
0123456789

Do
   Input "Entrer une chaîne ?";S$
   Input "Nombre de caractères ";N
   Print Left$(S$,N)
Loop
```

La deuxième variante de LEFT$ remplace les n caractères d'extrême gauche de la chaîne de destination par les caractères équivalents de *s$* Exemple:

```
A$="**** Basic"
Left$(A$,4)="AMOS"
Print A$
AMOS Basic
```

=RIGHT$= *(Renvoyer les caractères d'extrême droite d'une chaîne)*

d$=RIGHT$(s$,n)
RIGHT$(d$,n)=s$

RIGHT$ copie les *n* caractères de *s$* à *d$* en partant de la droite. Exemples:

```
Print Right$("AMOS Basic",5)
Basic
A$=Right$("0123456789ABCDEF",10)
Print A$
6789ABCDEF

Do
   Input "Entrer une chaîne?";V$
   Input "Ecrire le nombre de caractères?";N
   Print Right$(V$,N)
Loop
```

Comme avec LEFT$, il existe également une deuxième version de RIGHT$ dont la syntaxe est à l'image d'une instruction Basic.

RIGHT$(d$,n)=s$

Cette fonction transfère les n caractères se trouvant à l'extrême droite de la chaîne source *s$* dans la chaîne de destination *d$*. Tout caractère exédentaire dans *s$* sera rejeté. Exemple:

```
A$="AMOS ****"
Right$(A$,5)="Basic"
Print A$
AMOS Basic
```

=MID$= *(Renvoyer une chaîne de caractères depuis l'intérieur d'une chaîne)*

d$=MID$(s$,p,n)
MID$(d$,p,n)=s$

La fonction MID$ renvoie la section du milieu de la chaîne contenue dans *s$*. p marque la position des caractères au début de la sous- chaîne et n contient le nombre de caractères devant être recherchés. Si une valeur *n* n'est pas spécifiée dans l'instruction, tous les caractères sont lus jusqu'à la fin de votre chaîne. Exemples:

```
Print Mid$("AMOS Basic",6)
Basic
Print Mid$("AMOS Basic",6,3)
Bas

Do
   Input "Entrer une chaîne?";V$
   Input "Taper la position de départ et le nombre de caractères?";S,N
```

```
    Print Mid$(V$,S,N)
  Loop
```

Il existe également une instruction MID$.

MID$(d$,p,n)=s$

La version MID$ transfère n caractères dans *d$* depuis la position p+1 de la chaîne *s$*. Si une valeur *n* n'est pas spécifiée directement, les caractères seront remplacés jusqu'à la fin de la chaîne source *s$*. Exemples:

```
A$="AMOS *****"
Mid$(A$,5)="Magic"
Print A$
```
AMOS Magic
```
Mid$(A$,5,3)="Bas"
Print A$
```
AMOS Basic

```
Do
   Input "Entrer une chaîne destination";V$
   Input "Entrer une sous-chaîne";T$
   Input "Ecrire la position de départ et le nombre de caractères";S,N
   Mid$(V$,S,N)=T$ : Print V$
Loop
```

=INSTR *(Chercher l'occurence d'une chaîne dans une autre chaîne)*

f=INSTR(d$,s$ [,p])

La fonction INSTR vous permet de rechercher toutes les occurences d'une chaîne à l'intérieur d'une autre. Elle est souvent utilisée dans des jeux d'aventures pour diviser une ligne entière de texte en ses commandes individuelles. Il existe deux variantes possibles de la fonction INSTR.

f=INSTR(d$,s$)

Cette fonction recherche la première occurence de s$ dans d$. Si la chaîne est trouvée, sa position sera alors directement renvoyée, sinon le résultat sera forcé à zéro. Exemples:

```
Print Instr("AMOS Basic","AMOS")
```
1
```
Print Instr("AMOS Basic","S")
```
4

```
Print Instr("AMOS Basic","AMIGA")
0

Do
   Input "Chaîne à chercher";D$
   Input "Chaîne à trouver";S$
   X=Instr(D$,S$)
   If X=0 Then Print S$;"Pas trouvé"
   If X<>0 Then Print S$;"Trouvé à la position";X
Loop
```

La recherche commence normalement depuis le premier caractère de la chaîne de caractères *(d$)*. La deuxième version de INSTR vous permet de tester une section spécifique de la chaîne à la fois.

p est ici la position de départ de votre recherche. Tous les caractères sont numérotés de gauche à droite en partant depuis zéro. *p* est donc compris entre 0 et LEN(s$). Exemples:

```
Print Instr("AMOS BASIC","S",0)
4
Print Instr("AMOS BASIC","S",5)
8
```

=UPPER$ *(Convertir une chaîne en caractères majuscules)*

s$=UPPER$(n$)

La fonction UPPER$ convertit tous les caractères de *n$* en caractères majuscules et copie le résultat dans *s$*. Exemple:

```
Print Upper$("AmOs BaSic")
AMOS BASIC
```

=LOWER$ *(Convertir une chaîne en caractères minuscules)*

s$=LOWER$(n$)

La fonction LOWER$ convertit tous les caractères de *n$* en caractères minuscules. Elle est particulièrement utile dans les jeux d'aventures, puisque vous pouvez convertir toutes vos données en un format standard qui est bien plus facile à interpréter. Exemples:

```
Print Lower$("AMOS Basic")
amos basic

Input "Continuer (Oui/Non)";ANSWER$ ANSWER$=Lower$(ANSWER$) : If
```

```
ANSWER$="non" Then Edit
Print "Continuer votre programme..."
```

=FLIP$ *(Inverser une chaîne)*

f$=FLIP$(N$)

La fonction FLIP$ inverse simplement l'ordre des caractères contenus dans *n$*. Exemple:

```
Print Flip$("AMOS Basic")
```
cisaB SOMA

=SPACE$ *(Intègrer des espaces dans une chaîne)*

s$=SPACE$(n)

Cette fonction produit une chaîne de n espaces et les place dans s$. Elle est souvent utilisée pour aérer une section de texte avant de l'afficher à l'écran. Exemple:

```
Print "Vingt" ; Space$(20); "espaces"
```

=STRING$ *(Créér une chaîne de a$)*

s$=STRING$(a$,n)

La fonction STRING$ renvoie une chaîne de *n* copies du premier caractère dans *a$*. Exemple:

```
Print String$("Tapis",10)
```
TTTTTTTTTTTTTTTTT

Notez que STRING$("",N) est identique à SPACE$(N).

=CHR$ *(Renvoyer un caractère Ascii)*

s$=CHR$(n)

Créé une chaîne contenant un seul caractère au code Ascii n. Exemple:

```
For I=32 To 255 : Print Chr$(I); : Next I
```

Notez que seuls les caractères au code compris entre 32 et 255 peuvent être en fait affichés à l'écran. L'utilisation des autres codes est interne et ces derniers font office de codes de commande. Référez-vous aux commandes texte telles que CUP$ pour plus de

précisions.

=ASC *(Renvoyer le code Ascii d'un caractère)*

c=ASC(a$)

La fonction ASC vous donne le code Ascii interne du premier caractère de la chaîne a$.
Exemple:

```
Print Asc("B")
66
```

=LEN *(Renvoyer la longueur de la chaîne)*

l=LEN(a$)

LEN renvoie le nombre de caractères mémorisés dans *a$*. Exemple:

```
Print Len("12345678")
8
```

Ne confondez pas cette fonction avec la fonction LENGTH utilisée pour calculer la longueur d'une banque de mémoire AMOS.

=VAL *(Convertir une chaîne en un nombre)*

v=VAL(x$)
v#=VAL(x$)

La fonction VAL convertit une liste de chiffres décimaux mémorisés dans x$ en un nombre. Si ce processus échoue, une valeur zéro sera renvoyée. Exemple:

```
X=Val("1234") : Print X
1234
```

STR$ *(Convertir un nombre en une chaîne)*

s$=STR$(n)

La fonction STR$ convertit une variable nombre entier en une chaîne. Ceci peut s'avérer très utile parce que certaines fonctions, telles que CENTRE ne permettent pas d'entrer des nombres faisant office de paramètres. Exemple:

```
Centre "Mémoire restante est "+Str$(Chip Free)+" Octets"
```

Ne confondez pas la fonction STR$ et la fonction STRING$

Les opérations de tableau

SORT *(Trier tous les éléments d'un tableau)*

SORT a(0)
SORT a#(0)
SORT a$(0)

L'instruction SORT ordonne le contenu d'un tableau en ordre croissant. Ce tableau peut contenir soit des chaînes, des nombres entiers ou des nombres en virgule flottante. Le paramètre a$(0) spécifie le point de départ de votre tableau. Il doit toujours être positionné sur le premier élément du tableau (élément zéro). Exemple:

```
Dim A(25)
P=0
Repeat
   Input "Entrer un nombre (0 Pour Stopper)";A(P)
   Inc P
Until A(P-1)=0 Or P>25
Sort A(0)
For I=0 To P-1
   Print A(I)
Next I
```

MATCH *(Rechercher dans un tableau)*

r=MATCH(t(0),s)
r=MATCH(t#(0),s#)
r=MATCH(t$(0),s$)

La fonction MATCH recherche la valeur s dans un tableau trié. Si cette recherche aboutit, r est doté du numéro d'indice approprié. Mais si la recherche échoue, le résultat est négatif. La valeur absolue de ce chiffre vous donne l'élément le plus proche du paramètre original de recherche.
Notez que seuls les tableaux à simple dimension peuvent être vérifiés de cette façon. Il vous faudra également trier le tableau avec la commande SORT avant d'appeler cette fonction. Exemple:

```
Read N
Dim D$(N)
For I=1 To N
   Read D$(I)
Next I
Sort D$(0)
```

```
Do
   Input A$
   If A$="L"
      For I=1 To N : Print D$(I) : Next I
   Else
      POS=Match(D$(0),A$)
      If POS>0
         Print "Trouve",D$(POS);"En Memoire";POS
      End If
      If POS<0 and Abs(POS)>N
         Print A$," Pas trouvé. Plus proche de ";D$(0-POS)
      End If
      If POS<0 and Abs(POS)>N
         Print A$,"Pa trouve. Plus proche de";D$(N)
      End If
   End If
Loop
Data 10,"Adams","Asimov","Shaw","Heinlien","Zelazny", "Foster","Niven"
Data "Harrison","Pratchet","Dickson"
```

Notez que la fonction MATCH pourrait être utilisée conjointement avec la fonction INSTR pour donner une routine puissante d'analyseur syntaxique. Elle pourrait être utilisée pour interpréter les instructions entrées dans un jeu d'aventures.

6 Graphiques

AMOS Basic vous offre tout ce dont vous avez besoin pour produire de fabuleux graphiques. Vous avez à votre disposition un jeu complet de commandes vous permettant de dessiner des rectangles, des cercles et des polygones.

Toutes les opérations sont effectuées pratiquement instantanément. Même ici, AMOS Basic a plus d'un tour dans son sac.

Vous pouvez vous servir des fonctions graphiques AMOS dans tous les modes graphiques de l'Amiga y **compris** le mode Hold and Modify (HAM). Il est donc possible de créer des images HAM à vous couper le souffle directement à l'intérieur de l'AMOS Basic!

En outre, vous n'êtes pas seulement limité à l'écran visible. Si vous avez créé une très grande zone de jeu, vous pourrez avoir accès à chaque partie de votre écran en utilisant les classiques routines de dessin. Ainsi, il est facile de produire un scrolling en arrière-plan comme celui rencontré dans tous les jeux d'arcade comme Defender.

Les couleurs

L'Amiga vous permet de visualiser 64 couleurs à l'écran. Ces couleurs peuvent être sélectionnées en utilisant les commandes INK, COLOUR et PALETTE.

INK *(Définir la couleur utilisée par les opérations de tracé)*

INK coul[,papier][,contour]

coul spécifie la couleur devant être utilisée pour toutes les opérations ultérieures de tracé. La couleur de chaque point à l'écran est sélectionnée dans un des différents *registres* de *couleur*. Chacun de ces registres peuvent être affectés d'un code couleur choisi parmi 4096 couleurs.

Bien que l'Amiga ne vous offre que 32 registres de couleurs, AMOS vous laisse le choix de sélectionner les codes couleur qui sont compris entre 0 et 63. Cette possibilité vous permet de profiter au maximum des couleurs disponibles dans les modes HAM et Half-Bright. Vous trouverez une explication détaillée de ces modes dans le chapitre Ecrans.

La couleur *papier* définit la couleur des motifs de remplissage de l'arrière-plan eux-mêmes produits par la commande SET PATTERN.

La couleur *contour* sélectionne la couleur de contour de vos barres et de vos polygones. Cette option peut être activée au moyen de la commande SET PAINT suivante:

```
Rem Trace au hasard des cadres de taille aléatoire
Set pattern 0 : Set Paint 1
Repeat
   C=Rnd(16): Ink 16-C,0,C
   X=Rnd(320)-20 : Y=Rnd(200)-20 : S=Rnd(100)+10
   Bar X,Y To X+S,Y+S
```

Until Mouse Key

Notez que vous pouvez omettre n'importe quel de ces paramètres coul, papier et contour. Intégrez simplement deux virgules sans *espace* aux emplacements appropriés dans l'instruction. Par exemple:

Ink,,5:Rem Définit la couleur de contour

COLOUR *(Affecter une couleur à un index)*

COLOUR index,$RGB

L'instruction COLOUR vous permet d'attribuer une couleur à chacun des 32 registres de couleurs de votre Amiga.

Index est le code couleur que vous souhaitez modifier et celui-ci peut être compris entre 0 et 31. Comme vous le savez sans doute, n'importe *quelle* couleur peut être créée en mélangeant certaines quantités de couleurs primaires: rouge, vert et bleu. La teinte de votre couleur est entièrement déterminée par les intensités relatives de ces trois composants.

Le hardware de l'Amiga vous permet de sélectionner le composant de chaque couleur à partir d'une palette de 16 intensités. Cette palette peut être utilisée pour produire 16x16x16 (4096) différentes couleurs. Ces couleurs sont habituellement spécifiées en notation hexadécimale (base 16).

Chiffre hexadécimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Décimale	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

L'expression $RGB comprend trois chiffres compris entre 0 et F. Chaque composant détermine la luminosité d'une des couleurs primaires, rouge (R), vert (V) ou bleu (B). La valeur du composant est proportionnelle à l'éclat de la couleur qui lui est associée. Aussi, plus les numéros de code sont élevés, plus la couleur résultante est brillante.
Voici quelques exemples de cette notation:

Composants	**Forme hexadécimale**	**Couleur résultante**
R=0 V=0 B=0	$000	Noir
R=F V=0 B=0	$F00	Rouge vif
R=8 V=0 B=0	$800	Rouge foncé
R=F V=F B=0	$FF0	Jaune
R=0 V=F B=0	$0F0	Vert
R=8 G=0 B=F	$F0F	Violet
R=F G=F B=F	$FFF	Blanc
R=6 G=6 B=6	$666	Gris

Si vous voulez que le jaune ait le code couleur numéro 5, vous devez taper l'instruction suivante:

```
Colour 5,$FF0
```

A l'exécution de cette instruction, les graphiques à l'écran remplis de la couleur numéro 5 vireront immédiatement au jaune. Notez que les modes HAM et Extra Half Bright utilisent ces indices quelque peu différemment. Voir le Chapitre 9 pour plus de précisions.

=COLOUR *(Lire l'affectation des codes couleurs aux indexes)*

c=COLOUR(index)

La fonction couleur s'empare d'un numéro d'index compris entre 0 et 31 et renvoie le code couleur qui lui a été préalablement attribuée.

index est simplement le code couleur dont la teinte peut être modifiée. Vous pouvez utiliser cette fonction pour produire une liste de paramètres couleur courants dans votre Amiga comme par exemple:

```
For C=0 to 15
   Print Hex$(Colour(C),3)
Next C
```

COLOUR BACK *(Définir la couleur de fond)*

COLOUR BACK $RGB

Cette instruction définit la couleur de fond de l'écran. Cette couleur remplit les zones de l'écran qui ne sont pas utilisées, telles que les zones du haut ou le bas. Exemple:

```
Colour Back Colour(0) : Rem Attribue une seule couleur à l'intégralité de l'écran.
```

La couleur de fond ne peut pas être modifiée. Elle peut seulement être actualisée lorsqu'AMOS recalcule la liste de copper. Nous avons donc prévu une petite procédure appelée FAD_ALL qui actualise graduellement les couleurs de fond lors d'une opération FADE.

Vous trouverez cette procédure dans le fichier Squash_Procs.AMOS.

PALETTE *(Définir les couleurs courantes à l'écran)*

PALETTE list of colours

L'instruction PALETTE est simplement une version un peu plus puissante de COLOUR. Au lieu d'entrer un code couleur à la fois, la commande PALETTE vous permet d'installer une toute nouvelle palette de couleurs à l'aide d'une seule instruction.

Toutefois, vous n'avez pas à configurer toutes les couleurs de la palette en même temps. N'importe quelle combinaison de couleurs peut être sélectionnée

individuellement. Par exemple:

```
Palette $100,$200,$300 : Rem Définit trois couleurs
```

Vous pouvez également changer les couleurs sélectionnées au milieu de votre liste comme suit:

```
Palette $200,,$400 : Rem Changer les couleurs 0 et 2
```

Il faut savoir que seules les couleurs de la palette qui sont spécifiquement définies par cette commande seront en fait changées. Toutes les autres couleurs gardent leur code original. Voici quelques exemples:

```
Palette 0,$F00,$0F0
Palette 0,$770
Palette 0,,$66
Palette 0,$1,$2,$3,$4,$5,$6,$7,$8,$9,$A,$B,$C,$D,$E,$F
```

La palette de couleurs est automatiquement chargée au début de votre programme en utilisant une liste de codes couleur par défaut. Ces paramètres peuvent être définis en utilisant une simple option du programme de configuration AMOS.

Cette commande peut être également sélectionnée pour définir les couleurs utilisées dans les modes Ham et Half-Bright. Ceux-ci permettent d'étendre la palette de couleurs existante et de produire des douzaines de couleurs supplémentaires à l'écran. Voir le chapitre 10 pour une explication détaillée.

Commandes des caractères semi-graphiques

GR LOCATE *(Positionner le curseur graphique)*

GR LOCATE x,y

Cette fonction définit la position du curseur graphique aux coordonnées *x,y* de l'écran. Le curseur graphique est utilisé comme point de départ implicite pour la plupart des opérations de tracé. Si vous ommettez de configurer les coordonnées des commandes telles que PLOT ou CIRCLE, les objets seront tracés à n'importe quelle position du curseur. Par exemple:

```
Gr Locate 10,10 : Plot,
Gr Locate 100,100 : Circle,,100
```

=XGR *(Renvoyer l'abscisse X du curseur graphique)*
=YGR *(Renvoyer l'ordonnée Y du curseur graphique)*

x=XGR
y=YGR

Ces fonctions renvoient les présentes coordonnées du curseur graphique. Par exemple:

```
Circle 10,100,100
Print Xgr,Ygr
```

PLOT *(Tracer un seul point)*

PLOT x,y [,c]

La commande PLOT est la fonction de tracé la plus simple prévue par AMOS Basic. Elle trace un point aux coordonnées *x,y* en utilisant la couleur *c*. La nouvelle couleur d'encre sera maintenant utilisée dans toutes les opérations de tracé ultérieures.

Si la couleur *c* n'est pas précisée dans cette instruction, le point sera tracé dans la couleur d'encre courante. Par exemple:

```
Curs Off : Flash Off : Randomize Timer
Do
   Plot Rnd(319),Rnd(199),Rnd(15)
Loop
```

Il est également possible d'omettre les coordonnées X ou Y de cette instruction. Le point est alors tracé à la position du curseur graphique.

```
Plot 100,100,4
Plot,150
Cls : Plot ,
```

POINT *(Renvoyer la couleur d'un point)*

c=POINT(x,y)

POINT renvoie l'index couleur d'un point aux coordonnées *x,y*. Par exemple:

```
Plot 100,100
Print "La couleur à 100,100 est ";Point(100,100)
```

DRAW *(Tracer une ligne)*

DRAW est une autre instruction très élémentaire. Son action est de tracer une simple ligne droite sur l'écran d'Amiga.

DRAW x1,y1 TO x2,y2

Trace une ligne entre les coordonnées *x1,y2* et *x2,y2.*

DRAW TO x3,y3
Trace une ligne depuis la position courante du curseur graphique jusqu'à la coordonnée *x3,y3.* Par exemple:

```
Colour 4,$707 : Ink 4
Draw 0,50 To 200,50
Draw To 100,100
Draw To 0,50
```

Voir également à POLYLINE, INK.

BOX *(Tracer un cadre à l'écran)*

BOX x1,y1 TO x2,y2

La commande BOX trace un rectangle vide sur l'écran de l'Amiga. *x1,y1* sont les coordonnées au coin supérieur gauche du rectangle et *x2,y2* sont les coordonnées au point diagonalement opposé. Par exemple:

```
Curs Off: Flash Off : Randomize Timer
Do
   Ink Rnd(15)
   X1=Rnd(320) : Y1=Rnd(200) : Box X1,Y1 To X1+Rnd(50),Y1+Rnd(50)
Loop
```

Voir également SET LINE, INK et BAR

POLYLINE
(Tracé multiligne)

L'instruction POLYLINE est très similaire à l'instruction DRAW à l'exception qu'elle trace plusieurs lignes à la fois. Elle est capable de produire des polygones complexes en une seule instruction.

POLYLINE x1,y1 TO x2,y2 TO x3,y3 ...
POLYLINE TO x1,y1 TO x2,y2 ...

Avec *x1,y1* = coordonnées du point 1, *x2,y2* = point 2 et *x3,y3* = point 3

POLYLINE trace une ligne entre chaque couple de coordonnées de votre liste. Ainsi la première ligne est tracée du point 1 au point 2, la deuxième du point 2 au point 3 et ainsi de suite.

Elle est équivalente aux lignes suivantes:

```
Draw x1,y1 To x2,y2
Draw To X3,y3
Draw To x4,y4
```

Voici une instruction simple qui trace un triangle à l'écran de l'Amiga:

```
Polyline 0,20 To 200,20 To 100,100 To 0,20
```

Voir également à SET LINE, INK et POLYGON.

CIRCLE *(Tracer un cercle)*

CIRCLE x,y,r

La commande CIRCLE trace un cercle de rayon r et de centre x,y. Par exemple:

```
Curs Off : Flash Off : Randomize Timer
Do
   Ink Rnd(15)
   X=Rnd(200) : Y=Rnd(100) : R=Rnd(90) : Circle X,Y,R
Loop
```

Comme avec les instructions précédentes, si vous omettez d'entrer les coordonnées dans cette commande, le cercle sera tracé à la position courante du curseur. Par exemple:

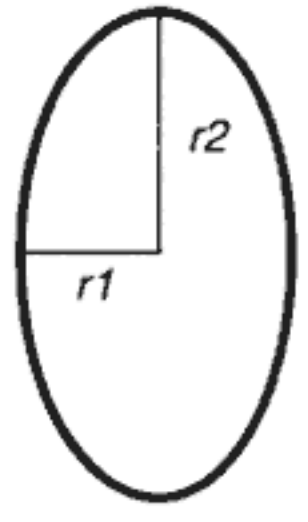
```
Plot 100,100 : Circle,,50
```

ELLIPSE *(Tracer une ellipse)*

ELLIPSE x,y,r1,r2

L'instruction ELLIPSE trace une ellipse aux coordonnées *x,y*. *r1* est le rayon horizontal. Il correspond à la moitié exacte de la largeur de l'ellipse. *r2* est le rayon vertical et il est utilisé pour calculer la hauteur de l'ellipse. La hauteur totale de l'ellipse est r*2.

Les rayon d'une ellipse



```
Curs Off : Flash Off : Randomize Timer
Do
   Ink Rnd(15) : X=Rnd(200) : Y=Rnd(100) : R1=rnd(90) : R2=rnd(90)
   Ellipse X,Y,R1,R2
Loop
```

Types de caractères semi-graphiques

AMOS Basic vous permet de tracer vos lignes en utilisant une gamme étendue de motifs de caractères semi-graphiques.

SET LINE *(Définir les styles de caractères semi-graphiques)*

SET LINE mask

La commande SET LINE configure le style de tous les caractères semi- graphiques utilisés par les commandes DRAW, BOX et POLYLINE.

Mask est un nombre binaire de 16 bits qui décrit l'apparence précise du tracé. Tous les points de la ligne devant être visualisée dans la couleur d'encre courante sont représentés par un "1" et ceux devant être définis dans la couleur de fond sont indiqués par un "0". Ainsi un trait normal est dénoté par un nombre binaire %1111111111111111 et sera ainsi visualisé:————————————.

Similairement, un trait tireté comme celui-ci: _ _ _ _ sera produit par un masque de %1111000011110000.

Les masques de trait peuvent recevoir des valeurs comprises entre 0 et 65533 et il est possible de produire une grande variété de types de traits. Par exemple:

```
Set Line $F0F0
Box 50,100 To 150,150
```

Ce style de trait est seulement applicable aux traits droits et ne s'applique pas aux formes tracées à l'aide des commandes CIRCLE ou ELLIPSE. Formes remplies

PAINT *(Remplissage)*

PAINT X, Y, mode

La commande PAINT vous permet de remplir n'importe quelle zone de l'écran de votre Amiga en couleur pleine. En outre, vous pouvez sélectionner un motif de remplissage de vos figures en utilisant la commande SET PATTERN.

x,y sont les coordonnées d'un point se trouvant à l'intérieur de la zone à remplir. Vous pouvez positionner le *mode* sur 0 ou 1. La valeur 0 termine l'opération de remplissage au premier pixel trouvé dans la couleur contour courante.

Le *mode* 1 arrête l'opération de remplissage lorsqu'il rencontre une couleur différente de celle existante.

La commande PAINT remplira toutes les surfaces que vous voulez à condition

qu'elles soient délimitées. Toutefois, si l'encadrement n'est pas complètement fermé, il laissera passer la couleur de remplissage dans le reste de l'écran. Voir **le fichier EXEMPLE 6.1** du dossier MANUEL pour une démonstration.

BAR *(Tracer un rectangle plein)*

BAR x1,y1 TO x2,y2

Trace une barre pleine depuis la position *x1,y1,* les coordonnées de l'angle supérieur gauche de la barre, à la position *x2,y2,* les coordonnées de l'angle diagonalement opposé. Par exemple:

```
Curs Off : Flash Off : Randomize Timer
Do
   X1=rnd(200) : Y1=rnd(100) : W=rnd(100) : H=rnd(80)
   Ink Rnd(15) : Bar X1,Y1 To X1+W,Y1+H
Loop
```

Voir également à BOX, SET PAINT and INK

POLYGON *(Tracer un polygone plein)*

POLYGON x1,y1 TO x2,y2 TO x3,y3 ...
POLYGON TO x1,y1 TO x2,y2 ...

L'instruction POLYGON produit un polygone plein dans la couleur d'encre courante. C'est en fait juste une version pleine de la classique commande POLYLINE. Comme auparavant, la couleur de remplissage est définie en utilisant l'instruction INK et le motif de remplissage en utilisant SET PAINT.

Les coordonnées *(x1,y1), (x2,y2), (x3,y3)* indique le premier et le dernier point des traits constituant le polygone. Il n'existe pas de vraie limite imposée au nombre de couples de coordonnées que vous pouvez utiliser, autre que la longueur maximale de tracé permise par AMOS Basic (255 caractères). Ceci signifie que vous pouvez créer des figures très compliquées au moyen de cette instruction.

Il existe également une deuxième variante de la commande POLYGON qui commence à tracer votre polygone à la position courante du curseur. Elle a le format suivant:

POLYGON TO x1,y2 TO x2,y2 ...

Mise à part les coordonnées de départ, elle est identique à la classique instruction POLYGON.

```
Do
   Ink Rnd(15)
   X1=Rnd(200): Y1=Rnd(100) : H=Rnd(100) : W=Rnd(90)
   Polygon X1,Y1 To X1+W,Y1 To x1+W/2,Y1+H To X1,Y1
```

Loop

Le programme ci-dessus permet de remplir l'écran de triangles joliment colorés. Voir également aux instructions POLYLINE, INK, SET PAINT.

Motifs de remplissage

En AMOS Basic, vous n'êtes pas seulement limité au remplissage en plein de vos figures. Des douzaines de motifs de remplissage vous sont proposés et vous pouvez même charger votre propre motif directement depuis la banque de sprites.

SET PATTERN *(Sélectionner un motif de remplissage)*

SET PATTERN pattern

Cette commande vous permet de sélectionner un motif de remplissage qui sera utilisé par vos opérations de tracé.

Il existe trois variantes:

pattern=0

C'est le motif par défaut, il remplit vos figures en plein dans la couleur de l'encre courante.

pattern>0

Si le numéro de motif de remplissage est supérieur à zéro, AMOS Basic sélectionne un des 34 styles de remplissage prédéfinis. Vous trouverez ceux-ci dans le fichier MOUSE.ABK de votre disquette de mise en route et vous pouvez les éditer au moyen de l'éditeur de sprites AMOS Basic. Notez que les trois premières images de ce fichier sont nécessaires pour afficher la flèche de la souris (voir CHANGE MOUSE). Les motifs de remplissage sont mémorisés dans les images à partir de la quatrième.

pattern<0

C'est l'option la plus puissante de toutes; l'option *pattern* se rapporte ici à une image de sprite dans la banque un. Le numéro d'image est calculé au moyen de la formule suivante:

SPRITE IMAGE = PATTERN *-1

L'image sélectionnée sera automatiquement tronquée avant sa visualisation, conformément aux règles suivantes.

- La largeur de l'image sera coupée à seize pixels
- La hauteur sera arrondie à la puissance la plus proche de deux, c'est-à-dire 1, 2, 4, 8, 16, 32, 64.

En fonction du type de votre image, le motif est tracé en monochrome ou en polychrome. Les images bichromes sont tracées en *monochrome.* Les couleurs réelles de votre image sont alors complètement ignorées et le motif est tracé en utilisant les couleurs courantes d'encre et de papier.

Il est également possible de produire des motifs de remplissage *polychromes.* Dans ce cas, les couleurs du premier plan de votre image sont mélangées à la couleur d'encre courante en utilisant un ET logique. Similairement, la couleur papier de votre motif est logiquement combinée avec le décor sprite (code couleur zéro). Si vous souhaitez utiliser les couleurs d'origine de vos sprites, il vous faudra définir les couleurs de l'encre et du décor de la manière suivante:

```
Ink 31,0
```

N'oubliez pas de charger la palette de sprites de la banque de sprites à l'aide de la commande GET SPRITE PALETTE avant d'utiliser ces instructions, sinon l'écran risque de sembler plutôt brouillé. Vous pouvez trouver des exemples de cette instruction dans **le fichier EXEMPLE 6.2** du dossier MANUEL. Voir les commandes CIRCLE, ELLIPSE, BAR et POLYGON.

SET PAINT *(Définir/redéfinir le mode contour)*

SET PAINT n

La commande SET PAINT sélectionne ou désélectionne le contour tracé par les instructions POLYGON ou BAR. Par défaut, ce mode est DESACTIVE.

Si *n=1*, ce mode contour sera activé et votre figure sera tracée dans la couleur contour spécifiée dans la commande précédente INK. Par exemple:

```
Ink,,5 : Set Paint 1 : Bar 100,100 to 200,150
```

Vous pouvez de nouveau désactiver ce mode en positionnant la commande SET PAINT sur zéro.

Les styles d'écriture

GR WRITING *(Changer le mode d'écriture)*

GR WRITING bitpattern

Lorsque vous tracez des graphiques à l'écran de votre Amiga, vous supposez évidemment que tout caractère se trouvant dessous soit recouvert. La commande GR WRITING vous propose quatre modes de tracé proposés. Vous pouvez les utiliser pour produire des douzaines d'effets incroyables.

bitpattern contient une suite de bits binaires spécifiant le mode graphique que vous souhaitez utiliser. Voici une liste des diverses possibilités ainsi qu'une brève explication

de leurs effets:

JAM1 mode *Bit 0=0*
JAM1 trace seulement les parties de vos graphiques définis dans la couleur **d'encre** courante. Toutes les sections tracées dans la couleur papier seront totalement omises. Utilisé avec la commande TEXT, ce mode est particulièrement utile puisqu'il vous permet d'intégrer directement un texte à un arrière-plan. Par exemple: ,

```
Ink 2,5 : Text 140,80,"Texte Normal" : Gr Writing 0 : Text 140,71,"JAM1"
```

JAM2 mode *Bit 0=1*
XOR combine vos nouveaux graphiques à ceux existants à l'écran en utilisant une opération logique connue sous le nom de OU exclusif.
Il a pour but de changer la couleur des zones d'une figure chevauchant une image existante.

Un autre effet intéressant du mode XOR est qu'il vous permet d'effacer tout objet de l'écran en sélectionnant simplement le mode XOR et en retraçant cet objet exactement à la même position. **Le fichier EXEMPLE 6.3** vous propose une simple démonstration de cette technique et produit un net effet de tracé élastique.

INVERSEVID *Bit 2=1*
Cette commande inverse l'image avant qu'elle soit dessinée. Ainsi, la couleur **d'encre** traçant les sections de votre image sera remplacée par la couleur papier courante et vice-versa. Le mode INVERSEVID est souvent utilisé pour créer un effet de vidéo-inverse.

Puisque ces modes sont définis en utilisant une configuration binaire, il est possible de combiner plusieurs modes.

```
Gr Writing 4+1 : Rem Définir JAM2 et INVERSEVID
Gr Wirting 4+2+1: Rem Choisit JAM2, INVERSEVID et XOR
Ink 2,5 : Text 140,80,"Texte Normal"
Gr Writing 5 : Text 140,71,"Inversevid+Jam2"
```

Remarque: Cette commande affecte seulement les opérations de tracé telles que CIRCLE, BOX et les caractères graphiques (TEXT). Le mode tracé utilisé par les commandes texte classiques comme PRINT et CENTRE est sélectionné au moyen d'une commande indépendante WRITING. Voir également les commandes AUTOBACK et WRITING.

CLIP *(Limiter le tracé des graphiques à une portion de l'écran)*

CLIP [x1,y1 TO x2,y2]

L'instruction CLIP restreint l'exécution de toutes les opérations de tracé à une zone rectangulaire de l'écran spécifiée par les coordonnées *x1,y1* et *x2,y2*.
x1,y1 représente les coordonnées du coin supérieur gauche du rectangle et *x2,y2* celles du coin inférieur droit.

Notez qu'il est parfaitement convenable d'utiliser des coordonnées se trouvant en dehors du cadre normal de l'écran. Toutes les opérations de coupure s'effectueront comme prévu, même si seule une portion du rectangle de coupure est en fait visible.
Vous trouverez un exemple détaillé de cette commande dans **le fichier EXEMPLE 6.4** du dossier MANUEL.

Comme vous pouvez le constater, seules les parties du cercle faisant partie du rectangle de coupure ont été tracées à l'écran. La zone de coupure peut être reconstituée dans la zone normale de l'écran en omettant les coordonnées de cette instruction.

Techniques avancées

SET TEMPRAS *(Définir la trame intermédiaire)*

SET TEMPRAS [adresse,taille]

Cette instruction permet aux programmeurs confirmés de l'Amiga de mesurer avec justesse la quantité de mémoire utilisée par les diverses opérations graphiques. **Attention!** Une utilisation incorrecte de cette instruction peut bloquer votre Amiga!

Lorsque un programme AMOS exécute une commande de remplissage, une zone-mémoire spéciale est réservée pour contenir le motif de remplissage. Cette mémoire est automatiquement renvoyée au système lorsque l'instruction est effectuée. La taille de la mémoire tampon est équivalente à un seul plan binaire en mode d'écran courant. Ainsi l'écran par défaut consomme un total de 8k.

La taille et l'emplacement du tampon graphique peuvent être changées à tout moment au moyen de l'instruction SET TEMPRAS.

La taille est le nombre d'octets que vous souhaitez réserver pour votre zone tampon. Elle est comprise entre *256* et *65536* octets.

La quantité de mémoire nécessaire pour tracer d'un certain objet peut être calculée de la façon suivante:

- Enfermez l'objet à tracer dans un rectangle.
- La zone nécessaire sera donnée par la formule: Taille=Largeur/8*Hauteur

Si vous avez l'intention d'utiliser la commande PAINT, assurez-vous que votre figure est bien *fermée*, sinon il vous faudra davantage de mémoire et le système peut se bloquer.

Le tampon peut être une adresse ou une banque. La mémoire que vous réservez pour ce tampon doit toujours être de la **mémoire chip**. Puisque la zone tampon est maintenant affectée une fois et pour toutes au début de votre programme, il n'est pas nécessaire de faire constamment des sauvegardes et de restaurer la mémoire tampon. L'exécution de vos programmes peut être ainsi 5% plus rapide.

Vous pouvez également réinitialiser la zone tampon en appelant la commande SET TEMPRAS sans avoir à recourir à des paramètres.

Référez-vous au programe **EXEMPLE 6.5** du fichier MANUEL pour une démonstration de cette commande.

7 Structures des commandes

Les langages de programmation les plus modernes comprennent une série d'instructions vous permettant de prendre des décisions et de réaliser des boucles dans vos programmes. Ces instructions sont connues sous le nom technique de structures de commandes. AMOS vous propose un jeu bien complémentaire des structures de commandes Basic. Toutes celles que vous connaissez déjà, GOTO, FOR...NEXT sont supportées, et nous y avons ajouté de nouvelles et fascinantes variantes telles que la commande ON...EVERY.

GOTO *(Se brancher sur un nouveau numéro de ligne)*

Dans les premiers temps difficiles de l'informatique, GOTO était sans doute l'instruction la plus communément utilisée de toutes. De nos jours, cette instruction est un peu démodée et peut être remplacée par des commandes structurées telles que DO...LOOP et IF...ELSE...ENDIF. Celles-ci sont souvent bien plus faciles à lire et nous vous recommandons donc d'éviter l'utilisation de GOTO si possible.

L'action de GOTO est de changer le cours d'exécution du programme. Il existe trois variantes de la commande GOTO permises dans AMOS Basic.

GOTO label

label est une borne optionnelle placée au côté d'une ligne. Les noms d'étiquette sont définis en utilisant les deux points ":" de la manière suivante:

label:

Le nom de l'étiquette peut comprendre toute chaîne de caractères alphanumériques que vous souhaitez, y compris le tiret "-". Ce nom est formé en utilisant les règles s'appliquant aux noms de variables et de procédures.

GOTO numéro ligne

Toute ligne AMOS Basic peut être précédée d'un nombre. Ces numéros de ligne sont seulement intégrés à des fins de compatibilité avec les autres versions du Basic (tels que STOS pour l'Atari ST). Il est préférable de fier sur les étiquettes puisque celles-ci sont bien plus faciles à lire et à se rappeler.

GOTO variable

Variable peut être n'importe quelle expression autorisée dans l'AMOS Basic. Cette expression peut être représentée par un nombre entier normal ou une chaîne. Les nombres entiers renvoient un numéro de ligne à votre GOTO, tandis que les chaînes contiennent le nom d'une étiquette.

Techniquement parlé, cette construction est connue sous le nom de goto calculé. Elle est en général désapprouvée par les programmeurs sérieux, mais elle peut s'avérer être quelques fois incroyablement utile. Exemples:

```
PLACE=3
BEGIN:
Goto "PLACE"+Str$(PLACE)-" "
End
PLACE3:
Print "Place trois!"
Goto BEGIN
```

Voir également ON GOTO

GOSUB *(Se brancher sur une sous-routine)*

GOSUB est une autre instruction démodée vous permettant, avec adresse, de diviser un programme en plus petits morceaux pouvant être mieux gérés et connus sous le nom de sous-routines. GOSUB a été presque entièrement remplacé par le système de procédure d'AMOS Basic. Toutefois, GOSUB est une étape pratique à mi-chemin lorsque vous effectuez des conversions de programmes à partir d'une autre version du Basic telle que le STOS.

Comme avec GOTO, il existe trois différents variantes de l'instruction GOSUB.

GOSUB n	Permet le branchement sur la sous-routine sur la ligne n.
GOSUB nom	Permet le branchement sur une étiquette AMOS.
GOSUB exp	Permet le branchement sur une étiquette ou sur une ligne résultant de l'expression exprimée dans exp.

```
Exemple:
  For I=1 To 10
  Gosub TEST
Next I
Direct
TEST:
Print "C'est un exemple de GOSUB" : Print "I égale ";I
Return : Rem Sortir de la sous-routine TEST et repasser la main au programme principal
```

Prenez la bonne habitude de placer toujours les sous-routines à la fin de votre programme principal comme ceci facilite leur saisie depuis les listages de programme. Il est également bon d'ajouter une instruction comme Edit ou Direct à la fin de votre programme principal, sinon AMOS peut tenter d'exécuter vos GOSUBs lorsque le programme est terminé et produire un message d'erreur.

RETURN *(Se débrancher d'une sous-routine appelée par une commande GOSUB)*

RETURN

RETURN permet de sortir d'une sous-routine qui a été préalablement appelée en utilisant GOSUB et de se rebrancher immédiatement sur la prochaine instruction Basic après le GOSUB d'origine.

Notez qu'une seule instruction GOSUB peut contenir plusieurs commandes RETURN. Vous pouvez ainsi sortir à n'importe quel point de votre routine en fonction de la situation.

POP *(Supprimer tous les libellés RETURN après un GOSUB)*

POP

Il est normalement interdit de sortir d'une instruction GOSUB en utilisant le classique GOTO. Cela peut être quelques fois gênant, particulièrement si une erreur se produit: vous ne pouvez alors pas revenir à votre programme à l'endroit précis où vous l'avez quitté.

L'instruction POP supprime l'adresse de renvoi produite par votre GOSUB et vous permet de quitter la sous-routine au point désiré, sans avoir à exécuter d'abord l'instruction finale RETURN. Exemple:

```
Do
   Gosub TEST
Loop
AU_REVOIR:
Print "Juste sorti"
Direct: Rem séparer les sous-routines de votre programme principal
TEST:
Print "Salut!"
If Mouse Key Then Pop : Goto AU_REVOIR
Return
```

Voir ON GOSUB

IF...THEN...[ELSE] *(Choisir entre diverses actions possibles)*

L'instruction IF...THEN vous permet de prendre de simples décisions dans un programme Basic. Sa syntaxe est la suivante:

IF conditions THEN bloc 1 [ELSE bloc 2]

conditions peut être une liste de tests comprenant des AND et OU. bloc 1 et bloc 2 doivent être des blocs d'instructions en AMOS Basic. Si vous voulez faire un branchement sur un numéro de ligne ou sur une étiquette, il vous faudra intégrer une

commande indépendante GOTO de la manière suivante:

If test Then Goto Label: Rem d'accord

Si vous essayez d'omettre cette instruction GOTO comme dans le Basic normal, AMOS considérera votre étiquette comme un nom de procédure et vous obtiendrez un message d'erreur procédure non définie.

If test Then Label: Rem Ceci appelle une PROCEDURE

La portée de cette instruction IF...THEN est limitée à une seule ligne de votre programme Basic. Elle a été remplacée par une commande bien plus puissante IF...ELSE...ENDIF.

IF...[ELSE]...ENDIF *(Test structuré)*

Bien que la forme d'origine du IF...THEN soit sans aucun doute pratique, elle est plutôt démodée aux côtés des possibilités offertes par une version vraiment moderne du Basic telle que l'AMOS. Cette instruction vous permet d'exécuter des blocs entiers d'instructions en fonction du résultat d'un seul test.

```
IF tests=TRUE
    Bloc d'instructions1
    : : :
ELSE
    Bloc d'instructions2
    : : :
ENDIF
```

Le bloc d'instructions peut représenter n'importe quel des groupes d'instruction AMOS Basic que vous souhaitez, y compris les autres commandes IF...ENDIF. Toutefois, il est **interdit** d'utiliser une instruction normale IF...THEN à l'intérieur d'un test structuré. Celles-ci devraient être remplacées par leur instruction équivalente IF...ENDIF comme suit:

If test Then Goto Label Else Label2

devient maintenant:

If test : Goto Label : Else goto Label2 : Endif

ou:

```
If test
   Goto Label
Endif
```

Voici un exemple de l'instruction IF...ENDIF en action:

```
Input "Entrez les valeurs de a,b et c";A,B,C
If A=B
   Print "Egal"
Else
   Print "Différent";
   If A<>B and A<>C
      Print ", et C N'est Aussi Pas La Même!"
   End If
End if
```

Chaque instruction IF de votre programme **doit** être associé à une seule commande ENDIF puisque celle-ci informe AMOS Basic de l'exécution des blocs d'instructions devant être exécutés dans votre test.

Notez que "THEN" n'est pas du tout utilisé par cette forme d'instruction. Vous allez prendre un peu de temps à vous familiariser à ces instructions si vous n'avez pas déjà mis en pratique une des autres versions du Basic pour le Commodore Amiga.

Voir AND, OR, NOT, TRUE, FALSE.

FOR...NEXT *(Répéter une partie d'un code un certain nombre de fois)*

C'est une façon classique de répéter des sections de vos programmes Basic. La syntaxe de l'instruction est la suivante:

```
FOR index=premier TO dernier [STEP inc]
bloc d'instructions
: : :
NEXT [index]
```

Une commande FOR...NEXT répète votre bloc d'instructions un certain nombre de fois. index dispose d'un compteur dont l'évolution est permise par l'exécution de chaque boucle. Au début de la boucle, le compteur prend le résultat de l'expression dans premier. Les instructions entre le FOR et le NEXT sont alors exécutées.

inc est la valeur ajoutée au compteur après l'exécution de chaque boucle par l'instruction NEXT. Si elle est omise, la valeur de progression est forcée à 1. Lorsque NEXT a mis à jour ce compteur, il teste si la valeur courante est supérieure à l'expression dans dernier. Si c'est le cas, la boucle est immédiatement arrêtée, et Basic exécute l'instruction juste après le NEXT. Dans le cas contraire, la boucle est relancée depuis le début.

Notez que si inc est une valeur négative, la boucle est arrêtée lorsque le compteur marque une valeur inférieure à celle entrée dans premier. Ainsi, l'exécution de la boucle entière est inversée.

Une fois à l'intérieur de votre boucle, index peut être lu de votre programme comme une variable normale. Mais vous **n'êtes pas** autorisé à changer sa valeur de quelques

façons que ce soit sinon cette opération produira un message d'erreur.

Chaque instruction FOR de votre programme **doit** correspondre à une seule instruction NEXT. Vous ne pouvez pas utiliser des formes abrégées comme dans les autres langages Basic telles que NEXT R1,R1. Voici deux ou trois exemples simples de ces boucles.

```
Flash Off
For I=32 To 255:Print Chr$(I); : Next I
   For R1=20 To 100 Step 20
        For R2=20 To 100 Step 20
          For A=0 To 3
          Ink A
        Ellipse 160.100.R1,R2
     Next A
  Next R2
Next R1
```

Notez la façon dont nous avons placé les boucles FOR...NEXT à l'une dans l'autre. Ce procédé s'appelle l'emboîtement.

WHILE...WEND *(Répéter une section de code tant qu'une condition logique est vraie)*

La commande WHILE vous présente une méthode pratique de répétition d'une série d'instructions Basic jusqu'à ce qu'une certaine condition soit satisfaite.

```
WHILE condition
: :
bloc d'instructions
: :
WEND
```

condition peut être n'importe quel jeu de tests; il peut comprendre les instructions AND, OR et NOT. Une vérification est faite à chaque tour de la boucle. Si le test renvoie une valeur -1 (vraie), alors les instructions se trouvant entre le WHILE et le WEND seront exécutées, sinon l'exécution de la boucle est suspendue et le Basic passera à l'instruction suivante. Tapez l'exemple suivant:

```
Input "Ecrivez un nombre";X
Print "Compter jusqu'à 11"
While X<11
   Inc X
   Print X
Wend
Print "Boucle suspendue"
```

Le nombre d'exécution de la boucle WHILE est fonction de la valeur entrée dans la

routine. Si vous entrez un nombre supérieur à 10, la boucle ne sera jamais exécutée. WHILE exécutera donc seulement les instructions si la condition est vraie (TRUE) au début de votre programme.

REPEAT...UNTIL *(Répéter jusqu'à ce la condition soit satisfaite)*

REPEAT
: :
bloc d'instructions
: :
UNTIL condition

REPEAT...UNTIL est similaire à WHILE...WEND sauf que le test d'intégralité est effectué à la fin de la boucle et non au début. La boucle sera sans cesse exécutée jusqu'à ce que la condition spécifiée s'avère fausse (FALSE). Ainsi la boucle sera toujours effectuée au moins une fois dans votre programme. Exemple:

```
Repeat
   Print "AMOS Basic"
Until Mouse Key<>0
```

Comme avec WHILE...WEND, vous devriez toujours vous rappeler de faire correspondre chaque REPEAT à un UNTIL.

DO...LOOP *(Boucle continue)*

DO
: :
Bloc d'instructions
: :
LOOP

Les commandes DO...LOOP s'emparent d'un bloc d'instructions Basic et l'exécutent continuellement. Afin de sortir de cette boucle, il vous faudra utiliser une instruction spéciale EXIT ou EXIT IF.

```
Do
   Print "Salut!"
Loop
```

L'avantage de ce système réside dans sa structure qui est différente de celle intégrant les boucles GOTO souvent utilisées dans les premières versions du Basic. Prenez l'exemple suivant:

```
TEST:
Input "Un autre jeu (O,N)";AN$
```

```
If Upper$(AN$)="N" Then Goto AU_REVOIR
GAME : Rem appeller jouer une procédure de jeu
Goto TEST
AU_REVOIR:
End
```

C'est un type assez courant de routine mais il est guère facile à lire. Etudions maintenant une deuxième variante utilisant DO...LOOP.

```
Do
   Input "Un autre jeu (O,N)";AN$
   Exit If Upper$(AN$)="N"
   GAME : Rem appeller jouer une procédure de jeu
Loop
End
```

Espérons que vous conviendrez que la nouvelle routine est bien plus claire.

EXIT *(Sortir d'une commande DO...LOOP)*

EXIT [n]

La commande EXIT vous permet de sortir immédiatement d'une ou de plusieurs boucles de programmes créées avec les instructions FOR...NEXT, REPEAT...UNTIL, WHILE...WEND, ou DO...LOOP. Votre programme AMOS se branchera alors directement sur la prochaine instruction après la boucle courante.

n est le nombre de boucles que vous souhaitez quitter. Si vous omettez d'entrer ce nombre, alors vous pourrez seulement sortir de la boucle la plus centrale. Voici un exemple:

```
Do
   Do
      Print "Boucle centrale"
      If Mouse Key=1 Then Exit
      If Mouse Key=2 Then Exit 2
      Wait 5: Rem Ralentir exécution de boucle pour la voir se dérouler
   Loop
   Locate 20, : Print "Boucle extérieure"
   Wait 5
Loop
```

EXIT IF *(Sortir d'une boucle en fonction d'un test)*

EXIT IF expression[,n]

Comme vous avez pu le constater dans l'instruction précédente EXIT, il est souvent

nécessaire de suspendre une boucle en fonction du résultat d'un ensemble spécifique de conditions. Cette opération est simplifiée au moyen d'une commande spéciale EXIT IF.

expression comprend une série de tests dans la syntaxe normale d'AMOS. EXIT est seulement permis si le résultat donné est -1 (vrai).

Le paramètre n vous permet de sortir de plusieurs boucles à la fois. Si vous oubliez de préciser ce paramètre, alors l'exécution de la boucle courante sera suspendue. Exemple:

```
While L=0
   Z=0
   Do
      Z=Z=1
      For X=o To 100
         Exit If Z=10,2 : Rem Sort de deux Boucles, DO et FOR
      Next X
   Loop
   Exit 1 : Rem Suspend While...Wend
Wend
```

EDIT *(Arrêter le programme et donner le contrôle à l'éditeur)*

EDIT

La commande EDIT arrête le programme courant et revient sur l'éditeur AMOS Basic. Elle peut s'avérer très utile lorsque vous mettez au point un de vos programmes.

DIRECT *(Sortir du mode Direct)*

DIRECT

DIRECT suspend l'exécution de votre programme et revient immédiatement sur mode Direct. Vous pouvez maintenant examiner le contenu de vos variables ou sortir vos programmes sur imprimante.

END *(Sortir du programme)*

END

L'instruction END vous fait sortir d'un programme. Il vous est maintenant donné l'option de revenir à l'éditeur ou au mode Direct. Appuyez sur la barre d'espacement pour éditer votre programme ou frappez la touche ESCAPE pour passer au mode Direct.

ON...PROC *(Se brancher sur une ou plusieurs des procédures en fonction d'une variable)*

ON v PROC proc1, proc2, proc3,...procN
Cette instruction permet de faire un branchement sur la procédure désignée en fonction du contenu de la variable v. Notez que les procédures que vous utilisez dans cette commande **ne peuvent pas** comprendre des paramètres. Si vous devez transférer des informations vers cette procédure, il vous faut les écrire sous la forme de variables globales. Voir le chapitre PROCEDURES pour une explication complète de cette technique.

La commande On...PROC est en fait équivalente aux lignes suivantes:

```
If v=1 Then Proc1
If v=2 Then Proc2
: : :
If v=n Then ProcN
```

ON...GOTO *(Se brancher sur l'une des lignes d'un bloc en fonction d'une variable)*

ON v GOTO ligne1, ligne2, ligne3...ligneN

L'instruction ON GOTO permet à votre programme de se brancher sur l'une des lignes en fonction du résultat de l'expression en v. Elle est équivalente aux lignes suivantes:

```
If v=1 Then Goto Ligne1
If v=2 Then Goto Ligne2
: : :
If v=n Then Goto LigneN
```

Pour que cette instruction puisse fonctionner, n doit être un nombre entier compris entre 1 et le nombre de destinations possibles. ligne peut être soit un numéro de ligne ou une étiquette. Voir GOTO, GOSUB, ON GOSUB.

ON...GOSUB *(GOSUB est une des routines d'un bloc en fonction de var)*

ON var GOSUB ligne1,ligne2,ligne3...

Elle est identique ON...GOTO sauf qu'elle utilise un gosub et non un goto pour se brancher sur une ligne. Lorsque l'exécution de la sous- routine est terminée, elle doit utiliser un RETURN pour se rebrancher sur la prochaine instruction après le ON...GOSUB.

Voir GOSUB et ON GOTO.

EVERY n GOSUB *(Appeler une sous-routine à intervalles réguliers)*

EVERY n GOSUB label

L'instruction ON EVERY appelle la sous-routine à une *étiquette* à intervalles réguliers, sans perturber votre programme principal.

n est la longueur de votre intervalle en 50ièmes de seconde. Le temps nécessaire à l'exécution de votre sous-routine doit toujours être inférieur à cette période, sinon vous obtiendrez un message d'erreur.

Après avoir lancé une sous-routine, le système sera automatiquement invalidé. Afin de réitérer cette sous-routine, il vous faut insérer une commande EVERY ON avant l'instruction finale RETURN. Voici un exemple:

```
Every 50 Gosub Test
Do
   Print At(0,0);"Boucle principale"
Loop
TEST:
Inc I : Print "C'est le numéro d'appel ";I
Every On:Return
```

Notez que l'instruction ON EVERY est similaire à l'instruction ON TIMER de l'Amiga Basic.

EVERY n PROC *(Appeler une procédure à intervalles réguliers)*

EVERY n PROC nom

EVERY PROC exécute automatiquement la procédure en question à intervalles réguliers au moyen d'un système d'interruption puissant.

n est la période d'attente entre chaque appel de procédure consécutif mesurée en unités de 50ième de seconde.

Comme avec la commande précédente, l'interruption doit être réactivée avant de quitter la procédure, sinon la routine sera appelée une seule fois. Vous devez donc utiliser EVERY ON avant de revenir sur votre programme principal avec END PROC.

```
Every 50 Proc TEST
Do
   Print At(0,0);"Boucle principale"
Loop
Procedure TEST
   Shared I
   Inc I : Print "C'est le numéro d'appel ";I
   Every On
End Proc
```

EVERY ON/OFF *(Alterner les appels de procédures automatiques)*

EVERY ON/OFF

EVERY ON redémarre le système d'interruption utilisé par les commandes EVERY. Elle doit être appelée juste avant la fin de l'exécution de la procédure ou de la sous-routine.

Similairement, EVERY OFF désactive complètement les appels. Elle est automatiquement exécutée au début de l'une de ces procédures.

BREAK ON/OFF *(Mettre en fonction ou hors fonction la touche Control+C ou touche BREAK)*

BREAK ON/OFF

Vous pouvez normalement interrompre un programme et revenir au Basic à tout moment en appuyant simplement sur deux touches Control et C. Cette fonction peut être désactivée au moyen de la commande BREAK OFF, ce qui protège votre programme contre sa reproduction de façon très rudimentaire. Vous pouvez évidemment réactiver les touches BREAK en utilisant BREAK ON.

Mais attention: Ne faites jamais fonctionner un programme protégé à moins que vous ayez d'abord effectué une copie de sauvegarde sur disquette. Sinon, si le programme est bloqué sur une boucle, vous pouvez facilement perdre quelques heures de travail.

Gestion des erreurs

ON ERROR GOTO *(Gérer une erreur dans un programme Basic)*

ON ERROR GOTO label

La commande ON ERROR vous permet de détecter et de corriger les erreurs à l'intérieur d'un programme AMOS Basic, sans avoir à revenir à la fenêtre de l'éditeur. Quelques fois, les erreurs peuvent surgir dans un programme et il est impossible de les prédire. Prenez la routine suivante comme exemple:

```
Do
   Input "Tapez deux nombres";A,B
   Print A;" divisé par ",B;" is";A/B
Loop
```

Ce programme fonctionne bien sauf si vous essayez de remplacer B par un zéro. AMOS Basic essaiera alors de diviser A par zéro ce qui vous donne un message indiquant une erreur de *division par zéro.*

Vous pouvez éviter ce problème en gérant l'erreur avec une instruction ON ERROR GOTO comme suit:

ON ERROR GOTO label

Si une erreur se produit dans votre programme Basic, AMOS se branche directement sur une étiquette. Ce sera le point de départ de votre propre routine de correction

d'erreur qui peut réparer l'erreur et vous permettre de vous rebrancher sur le programme principal sans danger.

Notez que vous **devez** faire sortir la sous-routine de gestion d'erreurs du programme au moyen d'une instruction spéciale RESUME. Vous n'êtes pas autorisé à vous rebrancher sur le programme en utilisant une instruction normale GOTO.

```
On Error Goto HELP
Do
   Input "Tapez deux nombres";A,B
   Print A;" divisé par ";B;" is ";A/B
Loop
Rem Sous-routine de gestion d'erreurs
HELP:
Print : Print : Bell
Print "Vous avez essayé de diviser"
Print "votre nombre par zéro"
Resume Next: Rem Redonner le contrôle à l'instruction suivante
```

Afin que ce système marche, il est important qu'une erreur ne surgisse pas dans votre routine de correction d'erreurs, sinon AMOS arrêtera votre programme ignominieusement.

L'action de ON ERROR GOTO peut être invalidée par l'appel de ON ERROR sans aucun paramètre.

```
On Error : Rem Arrêter la sous-routine de gestion d'erreurs
```

Vous pouvez également utiliser ON ERROR GOTO 0 à cette fin.

ON ERROR PROC *(Gérer une erreur en utilisant une procédure)*

ON ERROR PROC nom

Cette instruction sélectionne une procédure qui est appelée automatiquement si une erreur est détectée dans le programme principal. Elle est simplement une version structurée de l'instruction précédente ON ERROR GOTO.

Bien que cette procédure doive être terminée par le normal END PROC, il est nécessaire de revenir sur le programme principal à l'aide d'un appel supplémentaire pour permettre sa reprise (RESUME). Cette instruction peut être placée juste avant l'instruction finale END PROC. Voici un exemple:

```
On error Proc HELP
Do
   Input "Tapez deux nombres";A,B
   Print A;" divisé par ";B;" is ";A/B
Loop
Rem Sous-routine de gestion d'erreurs
```

```
Procedure HELP
   Print : Print : Bell
   Print "Vous avez essayé de diviser"
   Print "votre nombre par zéro."
   Resume Next : Rem Redonner le contrôle au Basic à l'instruction suivante
End Proc
```

Votre sous-routine de gestion d'erreurs peut aisément appeler toutes les procédures Basic que vous souhaitez. Chaque routine peut être protégée par sa propre sous-routine de gestion d'erreurs. Toutefois, il n'est pas possible de détecter des erreurs dans la sous-routine de gestion d'erreurs elle-même. Si un problème se présente dans la routine HELP, AMOS suspendra complètement l'exécution de votre programme.

RESUME *(Reprendre l'exécution du programme après une erreur)*

RESUME vous permet de revenir sur une instruction du Basic après que la sous-routine de traitement d'erreurs que vous avez créée à l'aide de l'instruction ON ERROR, ait corrigé le problème à sa source. Vous ne devez JAMAIS essayer d'utiliser GOTO dans ce contexte.

Il existe cinq syntaxes possibles de cette instruction:

RESUME

Permet le rebranchement sur l'instruction qui a causé l'erreur et l'essaie à nouveau.

RESUME NEXT

Permet le rebranchement sur l'instruction se trouvant juste après celle qui a produit l'erreur.

RESUME ligne

Donne le contrôle à la ligne spécifiée de votre programme principal. *ligne* peut se rapporter à une étiquette ou à un numéro de ligne normal. Elle NE doit PAS être utilisée pour relancer une procédure!

Les procédures sont considérées quelque peu différemmment. Si vous voulez faire un branchement sur une étiquette particulière, il vous faut placer une borne spéciale à un point de la procédure dans laquelle vous recherchez les erreurs éventuelles. Vous pouvez utiliser la commande RESUME LABEL à cet effet. Il existe deux variantes:

RESUME LABEL label

Définit l'étiquette qui doit être renvoyée après une erreur. Cette instruction doit être appelée en dehors de la sous-routine de gestion d'erreurs et juste après les premières instructions ON ERROR PROC ou ON ERROR GOTO.

RESUME LABEL
Utilisé à l'intérieur de votre sous-routine de gestion d'erreurs pour vous rebrancher directement sur l'étiquette que vous avez définie à l'aide de la commande précédente. Exemple:

```
On Error Proc HELP
Resume Label AFTER
Error 12
Print "Jamais imprimé"
AFTER : Print "Je me suis rebranché ici"
End
Procedure HELP
   Print "Oh là là, je pense qu'il y a une erreur!"
   Resume Label
End Proc
```

=ERRN *(Renvoyer le numéro de la dernière erreur)*

e=ERRN

Si vous avez créé vos propres routines de gestion d'erreurs en utilisant les commandes ON ERROR, il vous faudra pouvoir trouver précisément l'erreur qui s'est produite dans le programme principal.

Lorsqu'une erreur se produit, ERRN reçoit automatiquement son numéro d'identification. L'Annexe se trouvant à la fin de ce manuel vous donne une liste complète des erreurs possibles.

```
Print Errn
```

ERROR *(Produire une erreur et se rebrancher sur l'Editeur AMOS)*

ERROR n

La véritable action de la commande ERROR est de générer une erreur. Cela peut sembler plutôt idiot, mais c'est souvent bien utile.

Supposez que vous ayez créé une belle petite routine de gestion d'erreurs pouvant faire face à toutes les erreurs éventuelles du disque. ERROR vous offre une manière simple de simuler tous les divers problèmes sans être ennuyé par l'apparition de l'erreur réelle. Exemple:

```
Error 40
```

Permet de sortir du programme et affiche à l'écran un message d'erreur Etiquette non définie. Voici une autre syntaxe pratique de cette instruction:

```
Error Errn
```

Elle utilise la fonction ERRN pour afficher la situation courante de l'erreur après qu'un problème ait surgi dans votre programme.

8 Texte et fenêtres

Ce chapitre décrit les fonctions de fenêtrage et les fonctions texte supportées par l'AMOS Basic. Il existe des douzaines de commandes pouvant être utilisées pour réaliser tout ce que vous voulez, d'un simple tableau de marquage des points à une jolie boîte de dialogue.

Les attributs texte

PEN *(Sélectionner la couleur du texte)*

PEN index

L'instruction PEN sélectionne la couleur de l'intégralité du texte affiché dans la fenêtre courante. Vous pouvez choisir cette couleur parmi les 64 teintes proposées, en fonction du mode graphique sur lequel vous êtes. Par exemple:

```
For INDEX=0 To 15
Pen INDEX
   Print "Numéro stylo ";INDEX;At(20,);"Couleur"
Next INDEX
```

Par défaut, la couleur du pinceau est le numéro 2 de l'index (blanc).

=PEN$(n) *(Changer la couleur du stylo en utilisant les caractères de commande)*

a$=PEN$(n)

PEN$ renvoie une séquence de commandes spéciale qui change la couleur du stylo à l'intérieur d'une chaîne. L'affectation de la nouvelle couleur du stylo se fait automatiquement à l'affichage de cette chaîne à l'écran. Par exemple:

```
C$=Pen$(2)+"Blanc "+Pen$(6)+"Bleu"
Print C$
```

La syntaxe de la chaîne renvoyée par PEN$ se présente ainsi: Chr$(27)+"P"+Chr$(48+n).

Voir COLOUR, PALETTE, PAPER.

PAPER *(Définir la couleur de fond du texte)*

PAPER index

PAPER choisit la couleur de fond de votre texte. Comme avec PEN, index doit être un numéro de couleur compris entre 0 et 63. Par exemple:

```
Pen 2 : For INDEX=0 To 15
   Paper INDEX : Print "Numéro Papier ";INDEX;Space$(10)
Next INDEX
```

Normalement, le fond de l'écran prend par défaut le numéro de couleur un (orange). Référez-vous à la commande SCREEN OPEN qui vous donne une liste des diverses possibilités.

=PAPER$(n) *(Renvoyer une séquence de caractères pour sélectionner la couleur du papier)*

x$=PAPER$(index)

PAPER$ renvoie une chaîne de caractères qui change automatiquement la couleur de fond dès son affichage à l'écran. Par exemple:

```
Pen 1 : C$=Paper$(2)+ "Blanc "+Paper$(6)+"Bleu"
Print C$
```

Voir PEN, COLOUR, PALETTE.

INVERSE ON/OFF *(Entrer le mode vidéo-inverse)*

INVERSE ON
INVERSE OFF

La commande INVERSE inverse la couleur du texte et la couleur de fond sélectionnées par les commandes PEN et PAPER. Elle met le texte qui est affiché dans la fenêtre courante en vidéo-inverse.

INVERSE ON active le mode d'impression inversée. Similairement, ce mode peut être désactivé au moyen d'un simple appel à INVERSE OFF. Par exemple:

```
Print "Ce Texte Est En Mode Normal"
Inverse On : Print "Ce Texte Est En Vidéo-Inverse":Inverse Off
```

Voir SHADE, UNDER, WRITING.

SHADE ON/OFF *(Ombrer l'intégralité de tout nouveau texte)*

SHADE ON
SHADE OFF

SHADE ON met votre texte en sousbrillance en réduisant la luminosité des caractères à l'aide d'un masque. Vous pouvez retrouver votre texte d'origine en utilisant SHADE OFF. Exemples:

Shade On : Print "Texte ombré"
Shade Off : Print "Texte normal"

Voir UNDER, INVERSE, WRITING.

UNDER ON/OFF *(Sélectionner le mode souligner)*

UNDER ON
UNDER OFF

L'instruction UNDER ON souligne votre texte s'affichant à l'écran. Utilisez UNDER OFF pour désactiver ce mode. Par exemple:

Under On :Print "Souligné"
Souligné
Under Off:Print "Normal"
Normal

Voir SHADE, INVERSE, WRITING.

WRITING *(Changer le mode d'écriture du texte)*

WRITING w1 [,w2]

La commande WRITING vous permet de changer le mode d'écriture du texte pour effectuer d'autres opérations de traitement de texte. Elle détermine précisément la combinaison de votre nouveau texte aux les données existantes de l'écran.

La première valeur sélecte l'un des quatres modes d'écriture:

w1=0	REMPLACER (Défaut)	Votre nouveau texte écrasera les caractères existants.
w1=1	OU	Affiche les caractères à l'écran avec un OU logique.
w1=2	OU	exclusif Les caractères sont maintenant combinés avec ceux à l'écran en utilisant un OU exclusif
w1=3	ET	Introduit un ET logique entre le nouveau texte et le fond de l'écran.
w1=4	IGNORE	Toutes les opérations d'impression sont désormais totalement ignorées!

Le second numéro sélecte les parties du texte à afficher à l'écran. Cette option peut être omise si besoin.

w2=0	Normal	Le texte est affiché sur fond écran.
w2=1	Papier	Seul le fond du texte apparaît à l'écran.

w3=2 StyloIgnore la couleur du papier et écrit le texte sur un fond de couleur zéro.

Ne confondez **pas** cette commande avec GR WRITING.

Les fonctions du curseur

Si vous utilisez l'instruction PRINT, vos caractères seront toujours affichés à la position courante du curseur. AMOS contient une série de fonctions vous permettant de bouger ce curseur dans tout l'écran.

LOCATE *(Positionner le curseur)*

LOCATE x,y
LOCATE x,
LOCATE y,

LOCATE déplace le curseur texte vers les coordonnées x,y. Celui-ci fixe le point de départ de toutes les opérations d'impression ultérieures.

Toutes les positions d'écran sont spécifiées au moyen d'un ensemble spécial de *coordonnées texte*. Celles-ci sont mesurées en unités de caractères se rapportant à l'angle supérieur gauche de la fenêtre du texte. Par exemple, les coordonnées *15,10* se rapportent à un point se trouvant 10 caractères plus bas et 15 caractères plus loin sur la droite.

La portée admissible de ces coordonnées varie en fonction des dimensions précises de votre fenêtre et de la taille de votre police de caractères. Si vous essayez d'afficher quelque chose en dehors de ces limites, vous ferez apparaître un message d'erreur.

Notez que l'écran courant est toujours considéré comme une fenêtre 0. En fait, vous n'avez donc pas à ouvrir une fenêtre avant d'utiliser une de ces fonctions. Par exemple:

```
Locate 10,10 : Print "Salut!"
```

Si vous voulez positionner le curseur sur la ligne courante, vous pouvez omettre l'ordonnée. Par exemple:

```
Print "Salut Score 100000"; : Locate 12, : Print "12345";
```

De la même façon, vous pouvez bouger le curseur verticalement sans affecter l'abscisse existante.

```
Clw : Locate, 10 : Print "Dixième Ligne";
```

Voir CMOVE, AT, XCURS, YCURS.

CMOVE *(Mouvement relatif du curseur)*

CMOVE w,h

CMOVE déplace le curseur à une distance fixe de sa position présente. L'instruction s'effectue en ajoutant la valeur des variables *w* et *h* aux coordonnées courantes du curseur. Si la position du curseur est de 10,10, tapez:

```
Cmove 5,-5
```

qui déplace alors le curseur vers la position 15,5.

Comme avec LOCATE, vous pouvez omettre une des coordonnées si c'est possible. Par exemple:

```
Cmove ,2 : Rem Abaisser le curseur de deux positions
Cmove 2, : Rem Déplacer le curseur de deux positions vers la droite
```

=AT *(Renvoyer une séquence de caractères de commande pour positionner le curseur)*

x$=AT(x,y)

La fonction AT vous permet de changer la position du texte depuis le corps même d'une chaîne de caractères. Elle s'effectue en retournant une chaîne dont la syntaxe est la suivante:

```
Chr$(27)+"X"+Chr$(48+X)+Chr$(27)+"Y"+Chr$(48+Y)
```

A l'affichage de cette chaîne, le curseur du texte se déplace vers les coordonnées x,y. Par exemple:

```
a$="Ceci"+At(10,10)+"Est"+At(1,2)+"La Puissance de"+At(20,20) + "AMOS!"
Print A$
```

Ces commandes AT sont parfaites pour les tableaux de marquage des points car elles vous permettent de positionner votre texte une fois et pour toute pendant la phase d'initialisation de vos programmes. Vous pouvez alors actualiser le score à l'écran en utilisant une seule instruction d'impression. Voici un exemple:

```
HI_SCORE$=At(20,10)+"Salut Score"
SCORE=1000
Print HI_SCORE$;SCORE
```

Cette instruction est identique aux lignes suivantes:

```
SCORE=1000
Locate 20,10 : Print"Salut Score";SCORE
```

La première version est plus facile à modifier puisque vous pouvez déplacer le tableau de marquage en éditant simplement une seule chaîne, peu importe le nombre de fois qu'elle est utilisée dans vos programmes.

Voir LOCATE, CMOVE, CUP$, CDOWN$, CLEFT$ et CRIGHT$.

Fonctions de conversion

AMOS Basic vous offre quatre fonctions pratiques vous permettant d'établir facilement une conversion entre les coordonnées texte et les coordonnées graphiques.

=XTEXT *(Convertir une abscisse de grille graphique en une abscisse de grille texte)*

t=XTEXT(x)

Cette fonction prend une abscisse normale et la convertit en une abscisse texte se rapportant à la fenêtre courante. Si l'abscisse de l'écran se trouve en dehors de cette fenêtre, une valeur négative vous sera alors communiquée. Référez-vous au fichier **EXEMPLE 8.1.**

Voir YTEXT, LOCATE, WINDOPEN, XGRAPHIC, YGRAPHIC.

=YTEXT *(Convertir une ordonnée de grille graphique en une ordonnée de grille texte)*

t=YTEXT(y)

YTEXT convertit une ordonnée de grille d'écran standard en une ordonnée texte se rapportant à la fenêtre courante.

Voir XTEXT pour plus de précisions. Voir également YGRAPHIC, XGRAPHIC, LOCATE.

=XGRAPHIC *(Convertir une abscisse de grille texte en une abscisse de grille graphique)*

g=XGRAPHIC(x)

La fonction XGRAPHIC est en fait l'inverse de la fonction XTEXT en ce sens qu'elle prend une abscisse de texte comprise entre 0 et la largeur de la fenêtre courante et la convertit en une abscisse absolue d'écran. Elle est utilisée pour positionner le texte sur une zone graphique. Le fichier **EXEMPLE 8.2** du dossier MANUEL vous donne une démonstration de cette commande. Voir YGRAPHIC, XTEXT, YTEXT.

=YGRAPHIC *(Convertir une ordonnée de grille texte en une ordonnée de grille graphique)*

g=YGRAPHIC(y)

Cette fonction convertit une ordonnée de texte en une ordonnée d'écran absolue.

Voir XGRAPHIC, XTEXT, YTEXT.

Les commandes du curseur

Le curseur du texte sert de point de départ visible à toutes les opérations ultérieures de traitement de texte. Il est habituellement représenté par une barre clignotante horizontale, mais il peut être modifié au moyen des commandes SET CURS et CURS OFF.

En déplaçant le curseur à l'écran, vous pouvez positionner votre texte pratiquement où vous voulez. Rappelez-vous que toutes les mesures de coordonnées sont prises en utilisant les coordonnées de texte se rapportant à la fenêtre courante.

AMOS vous offre des douzaines de commandes simples vous permettant de positionner le curseur d'une façon précise à l'écran. Il est également possible de changer la forme physique du curseur du **texte** depuis votre programme Basic.

HOME *(Touche début d'écran)*

HOME

HOME déplace le curseur du texte vers l'angle supérieur gauche de la fenêtre courante (coordonnées 0,0). Par exemple:

```
Clw : Rem Effacer la fenêtre courante
Locate 10,10 : Rem Déplacer le curseur à la position 10,10
Print "Une démonstration de"
Home : Print "Début d'écran" : Rem Déplacer le curseur à la position 0,0
```

Voir LOCATE, XCURS, YCURS.

CDOWN *(Abaisser le curseur)*

CDOWN

CDOWN abaisse le curseur du texte d'une seule ligne. Exemple:

```
Print "Example" : Cdown : Cdown : Print "of cdown"
```

=CDOWN$ *(Renvoyer un caractère chr$(31)*

x$=CDOWN$

CDOWN$ est une fonction qui renvoie un caractère spécial de commande qui lui-même déplace automatiquement le curseur à l'impression d'un caractère. La fonction *Print CDOWN$* est donc identique à CDOWN. CDOWN$ vous permet de remplacer plusieurs

déplacements du curseur par une seule chaîne. Cette combinaison peut s'avérer quelques fois extrêmement utile. Par exemple:

```
C$="\"+Cdown$
For A=0 to 20
   Print C$;
Next A
```

Voir CUP, CLEFT, CRIGHT, AT.

CUP *(Monter le curseur)*

CUP

CUP monte le curseur du texte d'une ligne de la même façon que CDOWN la descend. Par exemple:

=CUP$ *(Renvoyer un caractère chr$(30)*

x$=CUP$

CUP$ renvoie une chaîne de commandes qui fait monter le curseur d'une seule position. Par exemple:

```
Print "Le curseur saute "+CUP$+"d'une line..."
```

Voir CLEFT, CDOWN, CRIGHT, AT.

CLEFT *(Curseur à gauche)*

CLEFT

L'instruction CLEFT déplace le curseur du texte d'une position vers la gauche. Par exemple:

```
Print "Example"; : Cleft : Cleft : Print "of cleft"
```

=CLEFT$ *(Produire une chaîne de caractère chr$(29) pour CLEFT)*

x$=CLEFT$
La fonction CLEFT$ renvoie un caractère de commande qui, à son affichage, exécute une opération CLEFT. Par exemple:

```
Print "Bonjour ";
Print Cleft$+Cleft$+"p";
```

Voir CUP, CRIGHT, CDOWN, AT.

CRIGHT *(Curseur à droite)*

CRIGHT

CRIGHT est la fonction opposée de CLEFT et elle permet de déplacer le curseur d'une position vers la droite.

```
Print "Example"; : Cright : Cright : Print "of cright"
```

=CRIGHT$ *(Produire une chaîne de caractère chr$(28) pour CRIGHT)*

X$=CRIGHT$

CRIGHT$ renvoie une chaîne de commandes qui exécute une opération CRIGHT à l'intérieur d'une suite de caractères. Par exemple:

```
Print Cright$:Rem Ceci a le même effet que CRIGHT
```

Voir CLEFT, CUP, CDOWN, AT.

XCURS *(Renvoyer l'abscisse du curseur du texte)*

x=XCURS

XCURS est une variable contenant l'abscisse courante du curseur du texte (en grille texte). Exemple:

```
Locate 10,0 : Print Xcurs

10
```

YCURS *(Renvoyer l'ordonnée du curseur)*

Y=YCURS

YCURS détient l'ordonnée du curseur du texte (en grille texte).

SET CURS *(Définir la forme du curseur du texte)*

SET CURS L1,L2,L3,L4,L5,L6,L7,L8

Cette instruction vous permet de changer la forme du curseur. La forme est spécifiée par une liste de configurations binaires contenues dans les paramètres *L1 à L18*. Chaque paramètre détermine l'apparence de la barre horizontale du curseur, numérotée

de haut en bas.

Chaque bit représente un seul point de la ligne courante du curseur. S'il est forcé à 1, le point sera alors tracé en utilisant la couleur numéro 3; sinon, il sera affiché dans la couleur courante du PAPIER. La meilleure façon de vous familiariser à cette instruction est de l'illustrer d'un exemple.

```
L1=%11111111
L2=%11111110
L3=%11111100
L4=%11111000
L5=%11110000
L6=%11100000
L7=%11000000
L8=%10000000
Set Curs L1,L2,L3,L4,L5,L6,L7,L8
```

Le curseur du texte doit normalement clignoter de façon continue. Pour supprimer cet effet, il vous suffit de faire un appel à la commande FLASH OFF avant d'utiliser cette instruction.

CURS ON/OFF *(Activer/désactiver le curseur du texte)*

CURS ON
CURS OFF

Cette commande cache ou réaffiche le curseur clignotant depuis la fenêtre courante. Elle n'a aucun effet sur les curseurs utilisés par d'autres fenêtres.

MEMORIZE X/Y *(Sauvegarder les coordonnées X ou Y du curseur du texte)*

MEMORIZE X
MEMORIZE Y

Les commandes MEMORIZE mémorisent la position courante du curseur dans un endroit sûr. Vous pouvez alors taper n'importe quel texte à l'écran sans détruire les coordonnées initiales du curseur. Celles-ci peuvent être rechargées au moyen des commandes REMEMBER.

REMEMBER X/Y *(Initialiser les coordonnées X ou Y du curseur du texte)*

REMEMBER X
REMEMBER Y

REMEMBER positionne le curseur aux coordonnées sauvegardées par le précédent appel à MEMORIZE. Si MEMORIZE n'a pas été utilisée, les coordonnées correspondantes seront forcées à zéro.

Un exemple de cette commande est cité dans le fichier EXEMPLE 8.3 du MANUEL.

CLINE *(Effacer une partie ou l'intégralité de la ligne courante du curseur)*

CLINE [n]

CLINE efface la ligne sur laquelle le curseur est positionné. Si *n* caractères sont présents, alors *n* caractères seront effacés depuis la position courante du curseur (sans le déplacer).

Entrez les lignes suivantes à partir de la fenêtre directe:

```
Print "Essai, Essai, Essai";
Cmove -7,
Cline 7
Cline
```

CURS PEN *(Choisir une nouvelle couleur pour le curseur du texte)*

CURS PEN n

Change la couleur du curseur du texte au numéro d'index *n*. Si le mode de votre écran vous offre un choix d'au moins quatre couleurs, alors le curseur prendra la couleur trois par défaut. Cette couleur est animée par clignotement implicite au chargement d'AMOS. Si vous choisissez une différente couleur, le curseur sera statique. Pour avoir un curseur qui clignote, il vous faut définir un nouvel ordre de couleurs en utilisant la commande FLASH.

Notez également que la nouvelle couleur s'applique seulement à la fenêtre couramment ouverte. Elle n'a aucun effet sur les curseurs utilisés dans les autres fenêtres. Par exemple:

```
Curs Pen 5
```

Voir FLASH, CURS ON/OFF.

Entrées/sorties du texte

CENTRE *(Centrer une ligne de texte à l'écran)*

CENTREa$

CENTRE saisit une chaîne de caractères exprimés en *a$* et l'affiche au centre de l'écran. Ce texte est toujours sorti sur la ligne courante du curseur. Par exemple:

```
Locate 0,1
Centre "C'est un TITRE centré"
Cmove,3
```

```
Centre "Et celui-ci en est un autre"
```

=TAB$ *(Afficher la tabulation)*

x$=TAB$

TAB$ renvoie un caractère de commande connu sous le nom de TAB (Ascii9). A l'impression de ce caractère à l'écran, le curseur du texte immédiatement se déplace de plusieurs positions vers la droite. La distance de ce déplacement peut être définie au moyen de la commande SET TAB. Par défaut, l'espacement de la tabulation est réglé sur quatre.

SET TAB *(Changer la tabulation)*

SET TAB n

Cette fonction spécifie la distance de déplacement du curseur du texte à l'impression du prochain caractère TAB. Par exemple:

```
Home : Rem Déplacer le curseur aux coordonnées 0,0
Set Tab 5 : Rem Définir l'espacement de tabulation à 5
Print Tab$;"Salut"; : Rem Imprime Salut depuis la position 5,0
A$=Tab$+Tab$
Print A$;"les gars" : Rem Imprime le texte à la position 15,0
```

Voir TAB$, CRIGHT.

REPEAT$ *(Répéter une chaîne)*

x$=REPEAT$(a$,n)

La fonction REPEAT$ vous permet d'imprimer la même chaîne de caractères plusieurs fois de suite en utilisant une seule instruction PRINT.

Elle s'effectue en ajoutant une suite de caractères de commandes exprimés en une variable X$. A l'impression de cette chaîne, AMOS répète simplement *a$ n* fois à l'écran. Les valeurs possibles de n sont comprises entre 1 et 207. Vous pouvez trouver une démonstration complète de cette commande dans le fichier **EXEMPLE 8.4**. La syntaxe de cette chaîne de commandes est la suivante:

```
Chr$(27)+"RO"+A$+Chr$(27)+"R"+Chr$(48+n)
```

Les commandes texte perfectionnées

ZONE$ *(Configurer une zone autour d'une portion de texte)*

x$=ZONE$(a$,n)

La fonction ZONE$ entoure une portion de texte d'une zone que l'on nommera zone hors texte. Après avoir défini une de ces zones, vous pouvez rechercher les collisions éventuelles entre la zone et la souris en utilisant la fonction ZONE. Cette fonction vous permet de créer des boîtes de dialogue et des menus puissants sans avoir à recourir à des astuces de programmation compliquées.

a$* est une chaîne contenant le texte d'un des "Boutons" de votre boîte de dialogue. Ce "bouton" est normalement activé si vous tapez *x$ à l'écran.

n spécifie le nombre de zones hors texte devant être définies. Le nombre maximal de ces zones est fonction de la valeur préalablement spécifiée à l'aide de la commande RESERVE ZONE.

Le programme de **EXEMPLE 8.5** du dossier du MANUEL vous donne une démonstration de cette commande. La syntaxe de la chaîne de commandes est la suivante:

```
Chr$(27)+"ZO"+A$+Chr$(27)+"R"+Chr$(48+n)
```

Voir ZONE, SET ZONE, RESERVE ZONE, RESET ZONE, BORDER$.

BORDER$ *(Encadrer une portion de texte)*

x$=BORDER$(a$,n)

Cette fonction renvoie une chaîne de commandes chargeant AMOS de tracer un cadre autour d'une portion de texte. Elle est couramment utilisée en combinaison avec la commande ZONE$ pour réaliser de jolis "boutons" comme ceux rencontrés dans les fenêtres et les boîtes de dialogue.

n est le numéro de contour compris entre 1 et 16 et *a$* contient le texte devant être encadré. Le texte en *a$* est placé à la position courante du curseur; ne soyez donc pas surpris si vous obtenez d'étranges résultats s'affichant à la position 0,0. Par exemple:

```
Locate 1,1 : Print Border$("AMOS Basic",1)
```

Pour obtenir une zone hors texte encadrée, il vous faut utiliser une ligne comme la suivante:

```
Print Border$(Zone$("CLIQUER ICI ",1),2)
```

Cette ligne permet d'encadrer le texte par une zone numéro 1 et un encadrement 2. La suite de commandes renvoyée est la suivante:

```
Chr$(27)+"EO"+A$+Chr$(27)+"R"+Chr$(48+n)
```

Voir ZONE$, ZONE, BORDER.

HSCROLL *(Défilement horizontal du texte)*

HSCROLL n

Cette commande décale horizontalement l'intégralité du texte de la fenêtre couramment ouverte d'une seule position. *n* peut prendre les valeurs suivantes:

1 = Déplace la ligne courante vers la gauche
2 = Fait défiler l'intégralité de l'écran vers la gauche
3 = Déplace la ligne courante vers la droite
4 = Deplace l'écran vers la droite

Ne confondez pas cette commande avec l'instruction SCROLL qui déplace l'intégralité de l'écran.

VSCROLL *(Défilement vertical du texte)*

VSCROLL n

Cette commande décale verticalement le texte dans la fenêtre couramment ouverte d'une seule position.

1 = Le texte se trouvant sur la ligne du curseur est descendu.
2 = Le texte se trouvant sur la ligne du curseur ou celle du dessous est remonté.
3 = Seul le texte depuis le haut de l'écran jusqu'à la ligne du curseur est remonté.
4 = Le texte depuis le haut de l'écran jusqu'à la position du curseur est abaissé.

Des interlignes sont introduites pour remplir les vides créés par le défilement.

Fenêtres

Les commandes de fenêtrage AMOS vous permettent d'exécuter opérations de traitement de texte et vos opérations graphiques dans une seule portion de l'écran courant.

Les fenêtres AMOS peuvent être utilisées avec les commandes de zone hors texte pour produire des boîtes de dialogue efficaces telles que des sélecteurs de fichiers et des tableaux de marquage des points. La typique boîte de mise en garde, par exemple, peut être facilement produite par deux ou trois lignes de programme AMOS Basic.

WINDOPEN *(Créer une fenêtre)*

WINDOPEN n,x,y,w,h [,contour]

L'instruction WINDOPEN ouvre une fenêtre et l'affiche à l'écran de l'Amiga. Cette fenêtre sera utilisée pour toutes les opérations ultérieures de traitement de texte.

n est le numéro de la fenêtre à définir. AMOS vous permet de créer autant de

fenêtres qu'il y a de mémoire disponible. Le numéro zéro de la fenêtre est affecté par défaut à l'écran courant. N'essayez donc pas de ré-ouvrir cette fenêtre en utilisant WINDOPEN ou de la changer au moyen de WINDSIZE ou WIND MOVE.

x,y sont les coordonnées graphiques de l'angle supérieur gauche de votre nouvelle fenêtre. Puisque les fenêtres AMOS sont tracées en utilisant le blitter d'AMIGA, la zone fenêtre doit toujours être délimitée par une matrice de 16 par 16 pixels. Pour obtenir cet effet, les abscisses sont automatiquement arrondies au multiple le plus proche de 16. En outre, si vous avez prévu d'encadrer votre fenêtre, vos abscisses seront déplacées de huit positions. Cette augmentation permettra que la zone de travail de votre fenêtre commence toujours au bord exact de l'écran. Il n'existe aucune restriction quelle qu'elle soit en ce qui concerne les ordonnées.

w,h spécifient la taille des caractères de la nouvelle fenêtre. Ces dimensions doivent toujours être divisibles par 2.

Contour sélectionne le style de contour de votre fenêtre. Il existe 16 styles possibles aux valeurs comprises entre 1 et 16.

Les fenêtres peuvent également comprendre deux lignes de titre optionnels. Un titre est affiché horizontalement en haut de la fenêtre et un autre peut être ajouté à la base de cette dernière.

Les fenêtres AMOS peuvent contenir des caractères ou des graphiques, comme avec le système Intuition. Il existe également une puissante instruction WIND SAVE qui sauvegarde la zone écran à l'intérieur de vos fenêtres. Si vous déplacez une de ces fenêtres, ce qui s'y trouve dessous sera automatiquement retracé. Par exemple:

```
For W=1 To 3
   Windopen W,(W-1)*96,50,10,10,1
   Paper W+3 : Pen W+6 : Clw
   Print " Window ";W
Next W
```

Vous pouvez aller de l'une à l'autre de ces fenêtres en utilisant la commande WINDOW. Essayez de taper les instructions suivantes en mode direct.

```
Window 1 : Print "AMOS"
Window 3 : Print "en action!"
Window 2 : Print "Basic"
```

Vous pouvez toujours repérer la fenêtre active grâce au curseur clignotant, bien que ce dernier puisse être désactivé en utilisant la commande CURS OF au besoin.

WINDSAVE *(Sauvegarder le contenu de la fenêtre courante)*

WINDSAVE

La commande WIND SAVE vous permet de déplacer vos fenêtres dans tout l'écran sans altérer votre écran existant.

Une fois que vous avez activé cette fonction, toute fenêtre que vous ouvrez ultérieurement sauvegardera automatiquement l'intégralité du contenu des fenêtres se trouvant dessous. Cette zone sera retracée si vous fermez une fenêtre ou si vous

changez sa position.

Il est important de noter que cette option sauvegarde le contenu de la fenêtre courante et non celle que vous êtes en train de définir avec WIND OPEN.

Au début de votre programme, la fenêtre courante est l'écran implicite et demande une mémoire importante se montant à 32k. Si vous souhaitez sauvegarder l'arrière-plan se trouvant dessous la boîte de dialogue, la plus grande partie de cette mémoire sera complètement gâchée.

Pour contourner ce problème, il vous suffit de créer une fenêtre fictive de la taille que vous désirez et de la positionner sur la zone que vous souhaitez sauvegarder. Vous pouvez alors exécuter la commande WIND SAVE et continuer votre programme normalement.

Si vous appelez votre boîte de dialogue par la suite, la zone de dessous sera sauvegardée en tant que partie de votre fenêtre fictive. Elle réapparaîtra automatiquement après que votre boîte ait été enlevée.

BORDER *(Changer le cadre de la fenêtre de l'écran courant)*

BORDER n,papier,stylo

La commande BORDER vous permet de sélectionner le numéro de style *n* du cadre de la fenêtre courante. Ce cadre est tracé en utilisant un groupe de caractères configurés dans la police implicite. Vous pouvez donc créer vos propres styles de cadre au moyen de l'utilitaire définisseur de polices.

Les options *papier* et *stylo* vous permettent de choisir facilement les couleurs de votre cadre. Les numéros de cadre possibles sont compris entre 1 et 16.

Un des paramètres peut être omis de cette instruction; les commandes suivantes sont donc toutes parfaitement autorisées:

```
Border 2,,
Border 1,2,3
Border 2,,3 : Rem N'oubliez pas de remplacer les valeurs manquantes par des virgules
```

TITLE TOP *(Définir un titre en haut de la fenêtre courante)*

TITLE TOP t$

L'instruction TITLE TOP dispose le titre donné par la chaîne *t$* sur la ligne supérieure de la fenêtre courante. Ce titre s'affiche alors le long du haut du cadre de votre fenêtre. Seules les fenêtres encadrées peuvent être titrées de cette façon.

```
Windopen 5,1,1,20,10
Title Top "Fenêtre numéro 5"
Wait Key
```

TITLE BOTTOM *(Définir le titre en bas de la fenêtre courante)*

TITLE BOTTOM b$
Cette commande assigne la chaîne b$ à la ligne inférieure de la fenêtre courante.

```
Windopen 5,1,1,20,10
Title Bottom "Fenêtre numéro 5"
Wait Key
```

WINDOW *(Modifier la fenêtre courante)*

WINDOW n

WINDOW active la fenêtre n en tant que fenêtre courante. Cette fenêtre sera alors utilisée pour les opérations ultérieures de traitement de texte.

Si le système automatique de sauvegarde a été initialisé, cette fenêtre sera automatiquement retracée avec son contenu. Pour une démonstration, chargez **le fichier EXEMPLE 8.6** du dossier MANUEL.

=WINDON *(Renvoyer la valeur de la fenêtre courante)*

w=WINDON

WINDON renvoie le numéro d'identification de la fenêtre active courante. Exemple:

```
Windopen Rnd(12)+1,10,10,10,10
Print "Fenêtre numéro ";windon," Active"
```

WIND CLOSE *(Fermer la fenêtre courante)*

WIND CLOSE

La commande WIND CLOSE efface la fenêtre courante. Si vous avez préalablement appelé la commande WIND SAVE, cette fenêtre sera remplacée par les graphiques sauvegardés, sinon la zone sera complètement supprimée de l'écran.

```
Windopen 1,1,8,38,18,1 : Print "Appuyez sur une touche pour fermer"
Print "cette fenêtre"
Wait Key
Wind Close
```

WINDMOVE *(Déplacer une fenêtre)*

WIND MOVE x,y

WINDMOVE déplace la fenêtre courante vers les coordonnées *x,y* du graphique. Comme pour les définitions d'origine de la fenêtre, l'absisse sera arrondie au multiple le

plus proche de 16 pixels. Exemples:

```
Wind Save
Wind Open 1,0,2,30,10,1
Wind Save
For M=1 to 100
   Pen Rnd(15) : Paper(15) : Print: Centre " Déplacer la fenêtre!"
   Wind Move Rnd(30)+1,Rnd(100)+1
   Wait Vbl
Next M

Wind Save : Wind Open 1,0,2,12,5 : Wind Save
Curs Off : Paper 5 : Pen 0 : Clw
Print : Print " Déplacez la" : Print " Souris" : Print " Flèche"
BEGIN:
On Error Goto STP
LOP:
WX=X Screen(X Mouse) : WY=Y Screen(Y Mouse)
If OX=WX and OY=WY Then Goto LOP Else Wind Move WX,WY
OX=WX : OY=WY : Goto LOP
STP:
OX=WX : OY=WY : Resume BEGIN
```

Si la sauvegarde de la fenêtre a été activée, cette fenêtre sera alors réaffichée à la nouvelle position, sinon l'écran demeurera inchangé.

WINDSIZE *(Modifier la taille de la fenêtre courante)*

WIND SIZE sx,sy

Cette commande modifie la taille d'une fenêtre AMOS. Les nouvelles tailles, *sx* et *sy*, sont spécifiées en unités de caractères. *Sx* doit être divisible par deux. Référez-vous au fichier **EXEMPLE 8.8.**

Si vous avez préalablement appelé la commande WIND SAVE, le contenu initial de votre fenêtre sera retracé par cette instruction. Si la nouvelle fenêtre est plus petite que la fenêtre d'origine, les portions de l'image se trouvant en dehors de cette fenêtre seront perdues. De la même façon, si vous avez élargi votre fenêtre, la zone se trouvant autour de votre région sauvegardée sera remplie avec la couleur papier courante. Notez également qu'après l'exécution de la commande WINDSIZE, le curseur du texte se repositionne aux coordonnées 0,0.

CLW *(Effacer la fenêtre courante)*

CLW

CLW efface le contenu de la fenêtre courante et la remplit en plein avec la couleur

papier courante. Par exemple:

```
Rem Efface fenêtre numéro W
Procedure CLEAR_WIN[W]
        WIND_OLD=Windon
        Window W : Clw
        Window WIND_OLD
End proc
```

Barres de curseur

AMOS intègre trois instructions vous permettant d'afficher à l'écran une barre standard de curseur. Ces types de barre ne peuvent pas être manipulées directement avec la souris. Afin de créer une barre de curseur de travail, il vous faut écrire une petite routine Basic pour que cette barre puisse opérer dans votre programme principal. En raison de la puissance réelle du système AMOS, cette routine est extrêmement facile à accomplir et les résultats peuvent être très impressionnants, comme vous pouvez le constater en étudiant **le fichier EXEMPLE 8.8.**

HSLIDER *(Tracer une barre de curseur horizontale)*

HSLIDER x1,y1 TO x2,y2,total,pos,taille

Cette commande trace une barre de curseur horizontale allant de *x1,y1 à x2,y2. x1,y1* représente les coordonnées de l'angle supérieur gauche de la barre. *x2,y2* déterminent la position du point diagonalement opposé.

total est le nombre d'unités divisant la barre de curseur. Chaque unité représente un seul élément de l'objet que vous contrôlez avec ce curseur. Ainsi, dans la fenêtre de l'éditeur, total est affecté du nombre de lignes du programme courant.

La taille de chaque unité est calculée à partir de la formule suivante:

(X2-X1)/TOTAL

pos est la position de la barre de curseur à partir du début du curseur de cette barre; elle est mesurée dans les unités que vous avez spécifiées en utilisant total. taille est la longueur de la barre de curseur exprimée dans les unités précédentes. Exemples:

```
Hslider 10,10 to 100,20,100,20,5
Hslider 10,50 to 150,100,25,10,10
```

Le fichier EXEMPLE 8.9 du dossier du MANUEL vous donne une démonstration pratique du déplacement du curseur de l'une de ces barres.

VSLIDER *(Tracer une barre de curseur verticale)*

VSLIDER x1,y1 TO x2,y2,total,pos,size

VSLIDER est pratiquement identique à l'instruction précédente HSLIDER. Elle affiche un simple curseur de *x1,y1 à x2,y2. x1,x2 et x2,y2* déterminent la position et la taille de ce dernier. Exemples:

```
Vslider 10,10 to 20,100,100,20,5
Vslider 0,0 to 319,199,10,2,6
```

Vous trouverez un exemple supplémentaire d'une barre de curseur de travail dans **le fichier EXEMPLE 8.10.**

SET SLIDER *(Sélectioner les motifs de remplissage du curseur de cette barre)*

SET SLIDER b1,b2,b3,pb,s1,s2,s3,ps

Bien que cette commande semble incroyablement compliquée, elle est en fait plutôt simple. SET SLIDER remplit de couleurs et de motifs le curseur de cette barre elles-même créée à l'aide des commandes HSLIDER et VSLIDER.

b1,b2,b3 définissent les couleurs de contour, de papier et d'encre du fond de la boîte et *pb* choisit le motif de remplissage utilisé pour ces zones.

De la même façon, *s1,s2,s3* saisissent les couleurs de la barre de curseur et sp sélectionne le motif de remplissage de cette dernière.

bp et *sp* peuvent représenter le motif de remplissage que vous souhaitez. Comme auparavant, une valeur négative se rapporte à une image de sprite de la banque courante de sprites. Vous pouvez ainsi créer des barres à curseur fantastiques et colorées. Exemple:

```
Centre "<Appuyez sur une touche>" : Curs Off
Do
   B1=Rnd(15) : B2=Rnd(15) : B3=Rnd(15) : BP=Rnd(24)
   S1=Rnd(15) : S2=Rnd(15) : S3=Rnd(15) : SP=Rnd(24)
   Set Slider B1,B2,B3,BP,S1,S2,S3,SP
   Hslider 10,50 To 300,60,100,20,25
   Vslider 10,60 To 20,180,100,20,25
   Wait Key
Loop
```

Les différentes polices

Il existe deux types différents de polices dans AMOS: les polices de caractères et les polices graphiques. Les polices de caractères sont celles utilisées par les commandes PRINT et WINDOW. Les polices de caractères sont aussi connues sous le nom de groupes de caractères et chaque fenêtre d'AMOS Basic peut avoir son propre groupe. Les polices graphiques sont bien plus flexibles et offrent une plage plus étendue de styles:

Caractères graphiques

Votre ordinateur Amiga est capable d'afficher une panoplie impressionnante de styles de caractères. Le disque d'origine du Workbench comporte huit jolies polices de diverses tailles et bien d'autres polices sont aisément disponibles dans le domaine publique. Si vous êtes passé au Workbench 1.3, vous aurez également la possibilité de concevoir vos propres polices en utilisant le programme FED se trouvant sur les disquettes supplémentaires.

AMOS Basic vous donne ainsi un libre choix d'impression. Les caractères peuvent être imprimés à tout point de l'écran, dans n'importe quel jeu de caractère proposé.

Les polices AMOS peuvent être utilisées pour ajouter du piquant aux jeux les plus élémentaires. Elles sont bien précieuses pour produire des écrans d'accueil et des tableaux de marquage des points dans vos jeux. Ainsi, il vous est conseillé d'en profiter dans vos programmes AMOS Basic.

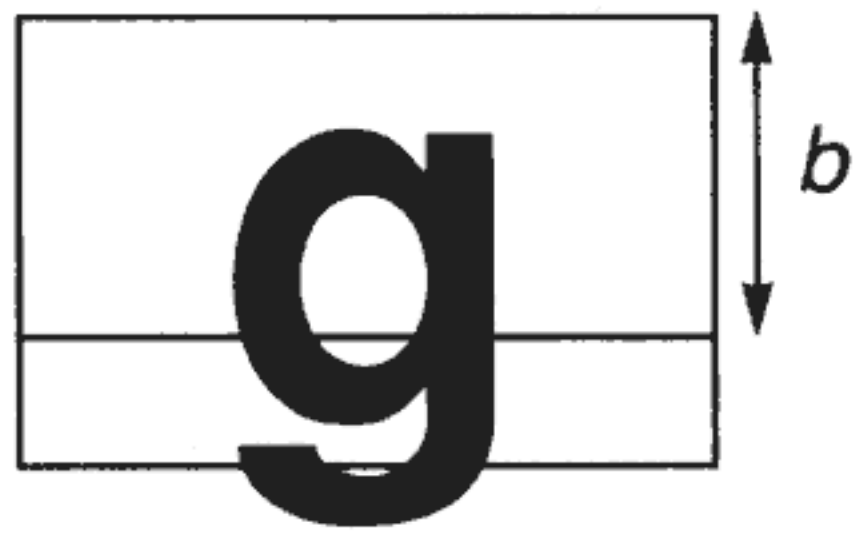
TEXT *(Imprimer des caractères graphiques)*

TEXT x,y,t$

TEXT imprime une ligne de texte en *t$* aux coordonnées graphiques *x,y*. Toutes les coordonnées sont mesurées en rapport à *la ligne* de base des caractères. Celle-ci peut être déterminée au moyen d'une fonction particulière TEXT BASE.

Normalement, la ligne de base est positionnée à la base du caractère mais certains caractères minuscules, tel le g, ont une queue qui s'étend légèrement en dessous de ce point, comme le montre le schéma suivant:

La base du caractère



Par défaut, le style de caractère est forcé à Topaz huit points. Vous pouvez changer ce style à votre convenance en utilisant l'instruction SET FONT. Essayez le programme suivant et constatez que le texte peut être placé à toute position pixel de l'écran.

```
Do
   Ink Rnd(15)+1,Rnd(15): Text Rnd(320)+1,Rnd(198)+1,"AMOS Basic"
```

Loop

Notez également que la couleur de votre texte est définie par INK et non par les commandes PEN ou PAPER comme vous pouviez le penser. Ceci met en valeur le fait que la commande TEXT est dans le fond une instruction graphique. Ainsi des séquences de commande générées par des fonctions telles que CUP$ seront imprimés à l'écran au lieu d'être correctement interprétées.

Il n'y a pas de retour à la ligne lorsque le texte atteint la fin de la fenêtre courante. Si vous essayez d'imprimer quelque chose de trop long à l'écran, le texte sera proprement coupé au bord existant de l'écran. L'exemple ci-dessous vous en propose une démonstration.

```
Print String$("A",100):Text 0,100,String$("A",100)
```

GET FONTS *(Créer une liste des polices disponibles)*

GET FONTS

La commande GET FONTS crée une liste interne de toutes les polices disponibles depuis la disquette courante de mise en route. Cette liste est nécessaire à l'exécution de la commande SET FONT. Ainsi, vous devez toujours appeler GET FONTS au moins une fois avant d'essayer de changer la présente définition de la police. Le contenu de cette liste peut être examiné au moyen la fonction FONT$.

Attention! Pour que la commande GET FONTS fonctionne, votre disquette courante de travail AMOS doit toujours contenir une copie du fichier standard LIBS. Il est important de se rappeler de ce fait lors de la distribution des programmes compilés ou seulement exécutés parce qu'à moins que les disquettes contiennent les fichiers requis, il est presque certain qu'AMOS Basic bloquera!

GET DISC FONTS *(Créer une liste de polices DISC)*

GET DISC FONTS

Cette commande est identique à l'instruction précédente GET FONTS sauf qu'elle recherche seulement les polices du disque. Ces polices sont contenues dans le fichier FONTS de votre disquette d'initialisation courante. Si vous voulez utiliser vos propres polices avec AMOS Basic, il vous faut les copier sur votre disquette de mise en route habituelle. Référez-vous au manuel livré avec votre Amiga pour mieux connaître cette procédure.

GET ROM FONTS *(Créer une liste des polices dans la mémoire permanente)*

GET ROM FONTS

La commande GET ROM FONTS vous présente une liste des polices intégrées dans la mémoire permanente de l'Amiga. Pour l'instant, il existe seulement deux de ces polices:

la huit points Topaz et la neuf points Topaz.

```
Get Rom Fonts
Set Font 1
Text 100,100,"Topaz 9"
Set Font 2
Text 100,120,"Topaz 8"
```

=FONT$ *(Renvoyer les caractéristiques des polices disponibles)*

a$=FONT$(n)

FONT$ renvoie une chaîne de 38 caractères décrivant le numéro de police *n*. Cette fonction vous permet d'examiner la liste de polices créées par l'appel précédent de l'une des commandes GET FONT. *n* est le numéro de police que vous souhaitez examiner.

a$* contient une liste de caractères représentant le nom et le type de votre police. Si une police n'existe pas, *a$ prendra une valeur nulle "", sinon une chaîne sera renvoyée dans le format suivant:

Caractère	Description
1-29	Nom de la police
30-33	Hauteur de la police
34-37	Identificateur (configurée sur Disque ou sur Mémoire permanente)

Vous trouverez un exemple de cette commande dans le fichier **EXEMPLE 8.11** du dossier MANUEL de votre disque programme AMOS. Il renferme un certain nombre de procédures utiles pouvant être aisément utilisées dans vos propres programmes.

SET FONT *(Choisir une police utilisée par l'instruction TEXT)*

SET FONT n

La commande SET FONT fait passer le groupe de caractères utilisés par la commande TEXT au numéro de police *n*. Si la police est conservée sur disque, elle sera automatiquement chargée dans la mémoire de votre Amiga. Tous les groupes précédents inutilisés seront simultanément supprimés.

Voici un exemple simple de cette commande en action.

```
Get Fonts
Set Font 2 : Text 100,100,"AMOS" : Set Font 1 : Text 100,120,"Basic"
```

Constatez que l'instruction GET FONTS est appelée avant de sélectionner la police. Ceci s'explique par le fait que SET FONT utilise la liste de polices créées par GET

FONTS. Une démonstration complète de la commande SET FONT est présentée dans le fichier **EXEMPLE 8.12.**

SET TEXT *(Définir le style de caractères)*

SET TEXT style
La commande SET TEXT vous permet de changer le style d'une police. Vous pouvez choisir entre trois *styles*: Souligner, mettre en caractères gras et en italique.
style est une configuration binaire se présentant dans le format suivant:

Bit	**Effet**
0	Force ce bit à un pour souligner vos caractères.
1	Sélectionne les caractères **en gras.**
2	Active le mode *italique.*

En configurant les bits appropriés dans cette configuration, vous pouvez choisir un des huit styles de caractères possibles. Voici un exemple que vous pouvez entrer dans votre ordinateur:

```
Colour 2,0 : Colour 1,$fff : Flash Off : Get Rom Fonts : Set Font 1
For S=0 To 7 : Set Text S : Text 100,S*20+20,"AMOS Basic": Next S
```

=TEXT STYLE *(Renvoyer le style de caractères courant)*

s=TEXT STYLE

Cette fonction renvoie le style de caractères préalablement définis par la commande SET TEXT. Le résultat représenté par s est un mode point du même format que celui utilisé par SET TEXT.

=TEXT LENGTH *(Renvoyer la longueur d'une portion de caractères graphiques)*

w=TEXT LENGTH(t$)

La fonction TEXT LENGTH renvoie la largeur en pixels de la chaîne de caractères *a$* de la police courante. La largeur d'un caractère varie en fonction de la taille de vos polices. En outre, les polices proportionnelles telles que Helvetica affectent des largeurs différentes à chaque caractère. Voici un exemple simple:

```
T$="Texte centré"
L=Text Length(T$)
Text 160-L/2,100,T$
```

=TEXT BASE *(Renvoyer la base courante du caractère)*

b=TEXT BASE

Cette fonction renvoie la position de la ligne de base de votre police. La *ligne de base* est le nombre de pixels imprimés à l'écran, nombre compris entre le sommet d'un caractère et sa base. C'est en fait similaire au point chaud d'un sprite ou d'un bob.

Comment installer de nouvelles polices

Si vo0us souhaitez utiliser vos propres polices dans AMOS Basic, il vous faut les sauvegarder sur une **copie** de votre disquette programme AMOS. La procédure de base est la suivante:

- Copiez les fichiers de police requis dans les POLICES: répertoire de votre disquette d'initialisation.

- Vous trouverez de plus amples renseignements dans le supplément de manuel livré avec la nouvelle version Workbench 1.3.

Dépannage

Bien que les polices soient assez faciles à utiliser, il existe deux ou trois pièges pour le novice. Voici une liste de solutions répondant à la plupart des problèmes courants.

Problème: GET FONTS semble ignorer toutes les polices du disque courant.
Solution: Vous avez sans doute retiré la disquette d'origine d'initialisation de votre lecteur implicite de disque. Les routines de bibliothèque de l'Amiga comptent trouver les POLICES: répertoire de votre disquette de démarrage. Vous pouvez changer l'affectation implicite du lecteur en utilisant le programme d'AFFECTATION dans le dossier UTILITAIRES. Voir GET DISC FONTS pour plus de précisions.
Problème:GET FONTS bloque l'Amiga.
Solution: Ce problème peut facilement survenir lorsque vous créez des programmes sous un format compilé ou seulement exécuté. GET FONTS a besoin de la bibliothèque DISCFONT dans le dossier LIBS pour s'exécuter. Si vous oubliez de copier ce dossier sur vos disquettes de distribution, vous obtiendrez un message d'erreur système qui peut bloquer votre Amiga.
Problème: La commande SET FONT renvoie une erreur *polices non examinées.*
Solution: Suppléez la commande GET FONTS d'un appel au début de votre programme. Il créera une liste de toutes les polices actuellement disponibles pouvant être utilisées par la commande SET FONTS.

9 Fonctions mathématiques

AMOS Basic offre une grande variété de fonctions mathématiques les plus couramment utilisées. Pour économiser de la mémoire, AMOS utilise les routines de bibliothèque standard de l'Amiga. Les bibliothèques appropriées seront automatiquement chargées du disque workbench au premier appel de l'une de ces fonctions dans une session particulière. Vous devriez donc vous assurer que la disquette courante contient le fichier MATHTRANS.LIBRARY du dossier LIBS.

Fonctions trigonométriques

Les fonctions trigonométriques mettent à votre disposition une panoplie d'outils mathématiques. Elles peuvent être utilisées à diverses fins, de l'enseignement à la création de formes d'ondes musicales complexes.

DEGREE *(Utiliser les degrés comme unités de mesure)*

DEGREE

Tous les angles sont généralement spécifiés en radians. Puisque les radians sont des unités de mesure d'angle plutôt difficiles à utiliser, il est possible de donner des instructions à AMOS pour qu'il accepte les degrés. Une fois que vous avez calculé cette fonction, tous les appels ultérieurs aux fonctions trig supposeront que vous utilisiez les degrés.

```
Degree
Print Sin(45)
```

RADIAN *(Utiliser les radians comme unités de mesure)*

RADIAN

La commande RADIAN informe AMOS que tous les angles doivent être entrés en utilisant les radians; c'est le défaut.

=PI# *(Une constante π)*

a#=PI#

Cette fonction renvoie le nombre appelé PI représentant le résultat de la division du diamètre d'un cercle par sa circonférence. PI est utilisé par la plupart des fonctions trigonométriques pour calculer les angles. Notez que le caractère # fait partie de la

désignation symbolique de la fonction. Il est utilisé pour éviter de les confondre avec vos propres noms de variables.

=SIN *(Sinus)*

s#=SIN(a)
s#=SIN(a#)

La fonction SIN calcule le sinus de l'angle en *a*. Notez que cette fonction renvoie toujours un nombre en virgule flottante. Exemple:

```
Degree
For X=0 To 319
   Y#=Sin(X)
   Plot X,Y#*50+100
Next X
```

Voir HSN

=COS *(Cosinus)*

c#=COS(a)
c#=COS(a#)

La fonction cosinus calcule le cosinus d'un angle. Normalement, tous les angles sont mesurés en radians. Vous pouvez changer cette unité de mesure en utilisant la commande DEGRE. Ajoutez les deux lignes suivantes à l'exemple ci-dessus, en vérifiant qu'elles sont bien insérées entre les instructions Plot et Next.

```
Y#=Cos(X)
Plot X,Y#*50+100
```

Voir ACOS, HCOS

=TAN *(Tangente)*

t#=TAN(a)
t#=TAN(a#)

La fonction TAN calcule la tangente d'un angle. Exemples:

```
Degree : Print Tan(45)
0.9999998
Radian : Print Tan(Pi#/8)
0.4141
```

Voir ATAN, HTAN

=ACOS *(Arc cos)*

c#=ACOS(n#)

La fonction ACOS prend un nombre compris entre -1 et +1 et calcule l'angle qu'il faudrait pour obtenir cette valeur avec COS.

Par conséquent, si X#=COS(ANGLE) alors ANGLE=ACOS(X#).

Notez que nous n'avons pas prévu une fonction ASIN parce qu'elle n'est pas vraiment nécessaire. Elle peut facilement être calculée en utilisant la formule suivante:

ASIN(X)=90-ACOS(X) : Rem Mesuré en degrés
ASIN(X)=1,5708-ACOS(X) : Rem en utilisant Radians

Exemple:

```
A#=Cos(45)
Print Acos(A#)
```

Voir COS, HCOS

=ATAN *(Arc tangente)*

t#=ATAN(N#)

La fonction ATAN renvoie l'arc tangente d'un nombre. Exemple:

```
Degree : Print(2)
0,03492082

Degree : Print Atan(0,03492082)
2
```

=HSIN *(Sinus hyperbolique)*

s#=HSIN(a)
s#=HSIN(a#)

HSIN calcule le sinus hyperbolique d'un angle *a*.

=HCOS *(Cosinus hyperbolique)*

c#=HCOS(a)

c#=HCOS(a#)
HCOS calcule le cosinus hyperbolique de l'angle *a*.

HTAN *(Tangente hyperbolique)*

t#=HTAN(a)
t#=HTAN(a#)

HTAN renvoie la tangente hyperbolique de l'angle *a*.

Fonctions mathématiques classiques

=LOG *(Logarithme)*

r#=LOG(v)
r#=LOG(v#)

LOG renvoie le logarithme en base 10 (log10) de l'expression en v#. Exemple:

```
print Log(10)
V#=Log(100)
```

=EXP *(Fonction exponentielle)*

r#=EXP(e#)

Calcule l'exponentielle de e#. Exemple:

```
Print Exp(1)
2,71828
```

=LN *(Logarithme décimal)*

r#=LN(#)

La fonction LN calcule le logarithme décimal ou népérien de l#. Exemple:

```
Print Ln(10)
2,30258

R#=Ln(100) : Print R#
4,60517
```

=SQR *(Racine carrée)*

s#=SQR(v)
s#=SQR(v#)
La fonction SQR calcule la racine carrée d'un nombre. Exemple:

```
Print Sqr(9)
3

Print Sqr(11,1111)
3,33333
```

=ABS *(Valeur absolue)*

r=ABS(v)
r#=ABS(v#)

La fonction ABS renvoie la valeur absolue de v, en ne tenant pas compte de son signe. Exemple:

```
Print Abs(-1),Abs(1)
1 1
```

=INT *(Convertir un nombre en virgule flottante en un nombre entier)*

i=INT(v#)

La fonction INT arrondit un nombre en virgule flottante en v au nombre entier inférieur le plus proche. Exemples:

```
Print Int(1.25)
1
Print Int(-1.25)
-2
```

=SGN *(Trouver le signe d'un nombre)*

s=SGN(v)
s=SGN(v#)

La fonction SGN renvoie une valeur représentant le signe d'un nombre. Trois cas se présentent:

-1 si v est négatif
0 si v a la valeur zéro
1 si v est positif

Créer des suites de nombres aléatoires

=RND *(Générateur de nombres aléatoires)*

v=RND(n)

La fonction RND produit un entier aléatoire compris entre 0 et n. Mais si n est inférieur à zéro, RND renverra la dernière valeur qu'elle a générée. Cette fonction peut s'avérer très utile lorsque vous mettez au point vos programmes. Exemple:

```
Do
   C=Rnd(15) : X=Rnd(320) : Y=Rnd(200) : Ink c : Text X,Y, "RANDOM"
Loop
```

RANDOMIZE *(Donner la valeur de départ d'un nombre aléatoire)*

RANDOMIZE seed

En pratique, les nombres produits par la fonction RND sont pseudo- aléatoires. Ils sont calculés à l'intérieur de la machine en utilisant une formule mathématique complexe. C'est la valeur *de départ* qui fixe le point de départ de l'exécution de ce calcul. Une valeur standard est donnée à cette valeur de départ lorsque vous chargez votre AMOS Basic dans l'ordinateur. Ainsi, la suite de nombres produite par RND sera exactement la même, chaque fois que vous lancez votre jeu!

Bien que l'utilisation de cette fonction convienne aux jeux d'arcade, elle vous causerait bien des problèmes si vous vouliez simuler un jeu de cartes tel que le Poker.

La commande RANDOMISER vous permet de résoudre ce problème en fixant directement la valeur de départ.

Vous pouvez attribuer toute valeur que vous souhaitez à La valeur de départ. Chacune produit sa propre suite de nombres.

Afin de produire de véritables nombres aléatoires, il vous faut varier la valeur de départ de vos jeux. Celle-ci peut être obtenue par l'instruction TIMER:

```
Randomize Timer
```

La fonction TIMER est une fonction Basic qui renvoie la durée pendant laquelle votre Amiga était allumé lors de la session courante. Toutes les durées sont mesurées en unités de 50ème de seconde.

Il est préférable d'utiliser cette instruction juste après que l'utilisateur ait entré des informations dans l'ordinateur. Vous pouvez simplement attendre le message d'abaissement d'une touche pour commencer le jeu puisqu'il est évidemment impossible de prédire à la seconde près le moment où l'utilisateur va frapper une certaine touche.

La suite de nombres produite par la fonction RND sera différente dans chaque jeu à condition que les valeurs de la fonction TIMER puissent être légèrement changées.

Manipuler les nombres

=MAX (Obtenir la plus grande des deux valeurs)

r=MAX(x,y)
r#=MAX(x#,y#)
r$=MAX(x$,y$)

La fonction MAX compare deux expressions et renvoie la plus grande valeur. Ces expressions peuvent être composées de nombres ou de chaînes de caractères, à condition que vous n'essayiez pas de mélanger les différents types d'expressions dans une seule instruction.

```
Print Max(10,4)
10
Print Max("Bonjour","Salut")
Hi
```

=MIN *(Renvoyer la plus petite des deux valeurs)*

r=MIN(x,y)
r#=MIN(x#,y#)
r$=MIN(s$,y$)

La fonction MIN renvoie la plus petite valeur des deux expressions. Celles-ci peuvent représenter des chaînes, des nombres entiers ou des nombres réels. Toutefois, vous devez seulement comparer les valeurs du même type. Exemple:

```
A=Min(10,4) : Print A
4
Print Min("Bonjour","Salut")
Bonjour
```

SWAP *(Intervertir la valeur de deux variables)*

SWAP x,y
SWAP x#,y#
SWAP x$,y$

Intervertit les données entre deux variables du même type, ce qui est équivalent à la ligne suivante:

```
DUMMY=X : X=Y : Y=DUMMY
```

Exemple:

```
A=10 : B=40 : Swap A,B : Print A,B
```

FIX *(Définir le degré de précision du résultat en virgule flottante)*

FIX(n)

La fonction FIX change la manière dont les nombres en virgule flottante seront sortis à l'écran ou sur imprimante. Quatre cas se présentent:
Si 0<n<16, alors n représente le nombre de chiffres se trouvant après la décimale.

Si n>16, l'impression sera proportionnelle et tous les zéros à droite seront supprimés.
Si n=16, alors le format sera celui du mode normal

Si n<0, alors tous les nombres en virgule flottante seront affichés sous un format exponentiel et la valeur absolue de n (ABS(n)) déterminera le nombre de décimales. Exemples:

```
Fix(2) : Print PI# : Rem Limite le nombre à deux décimale après la virgule.
Fix(-4) : Print PI# : Rem Force le mode exponentiel à quatre décimales après la virgule.
Fix(8) : Print PI# : Rem Revient sur le mode normal.
```

Attention: Cette fonction est tout à fait différente de la commande qui lui est équivalente en AMIGA Basic.

DEF FN *(Créer une fonction définie par l'utilisateur)*

DEF FN nom [(liste0]=expression

La commande DEF FN vous permet de créer vos propres fonctions au sein du programme AMOS Basic. Celles-ci peuvent être utilisées pour effectuer un calcul rapide et facile des valeurs dont vous avez couramment besoin.

nom est le nom de la fonction que vous souhaitez définir. liste est un groupe de variables séparées par des points-virgules. Seul le type de ces variables est important. Lorsque vous faites appel à votre fonction, toutes les valeurs entrées seront automatiquement substituées dans les positions correspondantes.

expression peut comprendre toutes les fonctions AMOS standard que vous souhaitez. Comme toutes les expressions Basic, elle est limitée à une seule ligne de votre programme.

Vous pouvez appeler la nouvelle fonction en utilisant l'instruction FN.

FN *(Appeler une fonction définie par l'utilisateur)*

FN nom [(liste variables)]

FN exécute une fonction définie en utilisant DEF FN. Voici deux ou trois exemples simples:

```
Def Fn N(X,Y)=X*Y
Print Fn N(2,3)

6

Def Fn SLICE $(A$,X,Y)=MID$(A$,X,Y)
Print Fn SLICE$("Bonjour",2;3)

onj
```

Remarquez la façon dont nous avons défini la fonction à l'aide de DEF FN avant de l'utiliser dans nos programme. C'est important!

10 Ecrans

Votre Commodore Amiga est capable d'afficher des images vraiment stupéfiantes. AMOS Basic vous permet d'intégrer directement ces images dans vos jeux à l'aide d'un arsenal impressionnant de commandes d'animation de l'écran.

Chaque mode graphique est exactement considéré de la même façon; il est donc aussi facile de créer un écran de 4096 couleurs Ham comme il l'est de produire un écran de 16 couleurs. Il est également possible de relier deux écrans indépendants en utilisant un système dual playfield. Ce système peut être exploité pour produire des effets parallaxes sensationnels, identiques à ceux rencontrés dans les jeux du commerce de haut niveau tels que Xenon II ou Silkworm!

L'écran implicite

Lorsque vous lancez un programme AMOS Basic, un écran implicite ou écran zéro est crée. Celui-ci représente un écran standard qui sera utilisé pour toutes les opérations normales de tracé.

Par défaut, le système sélectionne un écran de 16 couleurs et de dimensions 320 sur 200, pouvant être facilement modifiées depuis le corps de votre programme. En outre, vous pouvez également définir 7 autres écrans à l'aide de la puissante commande SCREEN OPEN.

DISPLAY HEIGHT *(Editer la hauteur possible de l'écran)*

=DISPLAY HEIGHT

Cette commande renvoie la donnée 311 en PAL et 263 en NTSC.

NTSC *(Indique le type d'écran en fonction)*

=NTSC

Cette commande renvoie VRAI si le système est en mode NTSC ou FAUX s'il est en PAL. C'est idéal pour le développement de logiciel à l'échelle internationale!

La musique est jouée à la même vitesse en PALM et en NTSC. Du fait que la vitesse de rafraîchissement de l'écran est coordonnée aux routines de musique, la musique doit être ralentie si elle est jouée sur un système NTSC. La vitesse de rafraîchissement de l'écran NTSC est de 60 fois par seconde tandis que celle de l'écran PAL est seulement de 50.

AMAL prend également en compte la routine d'interruption mais il n'est pas ralenti pour respecter les vitesses PAL. Vous devez donc prendre garde à ne pas faire synchroniser la musique et les animations en vous basant simplement sur la vitesse à laquelle elles se déroulent. Vérifiez que le programme est arrivé à une certaine image de l'animation ou que la musique a joué une certaine note. En utilisant cette technique,

vous vous assurez que le programme s'exécute aux points attendus sur tous les systèmes.

Faites attention si la synchronisation est très serrée dans vos programmes. Pour obtenir une animation de 50 images par seconde (60 images vous donnent une animation un peu plus rapide), le programme peut scintiller sur une télévision américaine!

Définir un écran

SCREEN OPEN *(Ouvrir un écran)*

SCREEN OPEN n,w, h, nc, mode

La commande SCREEN OPEN ouvre un écran et lui réserve de la mémoire. Le nouvel écran va maintenant servir de récepteur à toutes les opérations graphiques et de traitement de texte ultérieures effectuées dans votre programme.

n est le numéro d'identification de l'écran créé par cette instruction. Les valeurs possibles sont comprises entre 0 et 7. Si cet écran existe déjà, il sera totalement remplacé par votre nouvelle définition.

w représente la largeur de l'écran en pixels. Elle n'est pas limitée à la taille physique de votre écran. Vous êtes parfaitement autorisé à définir des écrans géants pouvant être manipulés au moyen de SCREEN OFFSET.

h détermine la hauteur de votre écran en utilisant le même système. Si vous disposez d'une mémoire suffisante, vous pourrez facilement créer des écrans virtuels qui sont bien plus grands que la zone visible de l'écran réel. Ces écrans peuvent être utilisés en même temps que toutes les manipulations d'écran habituelles. Ainsi, vous pouvez créer des images en dehors de la zone écran et les faire défiler pour les visualiser à l'aide de la commande SCREEN OFFSET.

nc détermine le nombre de couleurs requis pour le nouvel écran. La plage des couleurs disponibles est comprise entre 2 et 64(Extra Half Bright). Vous pouvez également accéder au mode Hold and Modify (Ham), qui est particulier à l'Amiga, comportant 4096 couleurs.

mode vous permet de choisir la largeur des points à l'écran. L'Amiga supporte des largeurs d'écran en 320 ou 640 pixels. Vous pouvez sélectionner la largeur requise en positionnant Mode sur LOWERES (0) ou HIRES ($8000).

Voici une liste d'options d'écran donnant une indication de la quantité de mémoire qu'elles consomment.

Colours	Résolution	Mémoire	Notes
2	320x200	8k	PAPIER=0 STYLO=1 Cursor=1, pas de clignotement
	640x200	16k	PAPER=0 STYLO=1 Curseur=1, pas de clignotement
4	320x200	16k	PAPIER=1 STYLO=2 Curseur=3, clignotement=3

Colours	Résolution	Mémoire	Notes
	640x200	32k	: : :
8	320x200	24k	PAPIER=1 STYLO=2 Curseur=3, clignotement=3
	640x200	48k	
16	320x200	32k	Utilisé par écran 0 (défaut).
	640x200	64k	
32	320x200	40k	
64	320x200	48k	Mode Extra Half bright
4096	320x200	48k	Mode Hold and Modify

Notez que la taille des mémoires de ce tableau s'appliquent seulement à un écran standard. Si vous créez des écrans plus hauts ou plus larges, la quantité de mémoire qui est consommée sera évidemment bien plus grande. L'écran zéro est équivalent à:

```
Screen Open 0,320,200,16,Loweres
```

Voici d'autres exemples illustrant les écrans AMOS:

```
Rem Ouvre un 640x200 écran haute res avec 8 couleurs
Screen open 1,640,200,8,Hires

Rem Ouvre un écran 2 en tant qu'écran HAM
Screen open 2,320,256,4096,Lowres

Rem Ouvre un écran 3 en tant que grand écran à 8 couleurs.
Rem Seuls les premiers 320x256 seront visibles à tout moment.

Screen open 3,500,400,8,Lowres
Rem Cette démo ouvre 8 écrans et les affiche tous!

Curs Off: Cls 13 : Paper 13
Print : Centre "Je suis écran 0 à l'arrière!"
For A=1 To 7
   Screen Open A,320,20,16,Lowres
   Curs off
   Cls A+5
   Paper A+5
   Centre "Je suis l'écran "+Str$(A)
   Screen Display A,,50+A*25,,8
Next A
Direct
```

Voir SCREEN OFFSET, SCREEN DISPLAY and VIEW

SCREEN CLOSE *(Effacer un écran)*

SCREEN CLOSE n

La fonction SCREEN CLOSE efface le nombre d'écran n et libère la zone de la mémoire qu'elle utilise pour le reste de votre programme.

AUTO VIEW ON/OFF *(Contrôler le mode de visualisation)*

AUTO VIEW OFF

Si vous ouvrez un écran en utilisant SCREEN OPEN, le nouvel écran s'affiche d'habitude immédiatement. Cette fonction peut s'avérer très utile lors des étapes d'initialisation de vos programmes.

La commande AUTO VIEW OFF vous permet de garder un contrôle total sur le processus d'actualisation. Elle met le système automatique de visualisation hors fonction. Vous pouvez alors actualiser les écrans à un moment pratique lors du déroulement de votre programme en utilisant l'instruction VIEW.

AUTO VIEW ON

Active l'actualisation automatique de l'écran. Dans ce mode, tout changement d'écran s'affiche immédiatement à votre écran de télévision.

DEFAULT *(Réinitialiser l'écran)*

DEFAULT

Ferme tous les écrans ouverts et fait reprendre à l'écran sa configuration initiale. Exemple:

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF",0
Wait Key
Default
```

VIEW *(Afficher les positionnements courants de l'écran)*

VIEW

La commande VIEW affiche les changements apportés aux définitions courantes de l'écran au prochain balayage du téléviseur. Vous devez seulement utiliser cette commande si AUTOVIEW est DESACTIVE.

Les modes graphiques particuliers

La couleur de chaque point de l'écran est déterminé par une valeur mémorisée dans un .

des 32 registres de couleur de l'Amiga. Chaque registre peut être chargé depuis une plage de 4096 différentes couleurs.

Bien que ce nombre de 32 couleurs puisse vous sembler plutôt grand, particulièrement selon les normes ST, il était insuffisant du point de vue des designers de l'Amiga. Ils auraient pu augmenter le nombre de registres de couleurs, mais cette solution, qui est la plus facile, fut rapidement écartée pour des raisons de coût.

Au lieu de cela, ils ont inventé deux modes graphiques particuliers qui ont habilement exploité les registres existants pour augmenter le nombre maximal de couleurs à l'écran.

Il se peut que vous ayiez déja rencontré ces modes. Ce sont les modes au nom barbare de Extra Half Bright et Ham. AMOS Basic vous prête un support complet dans l'utilisation de ces modes. Voici une brève explication.

Le mode Extra Half Bright (EHB)

Le mode Extra Half Bright double le nombre de couleurs possible à l'écran pour arriver à un total de 64. Il produit deux couleurs pour chacun des 32 registres de couleur possibles.

Les 32 premières couleurs saisissent le code couleur depuis l'un des registres. Chaque registre contient une valeur comprise entre 0 et 4095 déterminant la teinte précise de la couleur finale.

Le second groupe de couleurs, dont les nombres sont compris entre 32 et 63, saisit un des registres précédents et divise son contenu en deux. 32 couleurs supplémentaires sont ainsi produites et leur teinte est aussi lumineuse que les couleurs des registres ordinaires.

Supposez que la couleur zéro contienne une valeur de $FFF(Blanc). Le code couleur 32 serait alors affiché en utilisant une valeur de $777 (Gris). La teinte des couleurs comprises entre 32 et 63 peut être élaborée en utilisant la formule simple qui suit:

```
Colour N,Colour(N-32)/2
Rem Voici un exemple démontrant ce principe:
Screen Close 0
Screen Open 2,320,167,64,Lowres : Flash Off
For I=1 To 32
   Ink I
   Bar 0,(I-1)*5 To 160,(2+I-1)*5
   Ink I+32
   Bar 160,(I-1)*5 To 319,(2+I-1)*5
Next I
```

Afin d'exploiter pleinement le mode EHB, il est nécessaire d'attribuer les teintes les plus lumineuses de votre palette aux 32 registres. Ainsi, vous produirez automatiquement une liste de teintes intermédiaires dans les couleurs de 32 à 63.

Mise à part la palette de couleurs, le mode d'écran EHB est identique à n'importe quel autre mode d'écran. Il n'existe aucune restriction quelle qu'elle soit à son utilisation. Il est même possible de créer des Bobs dans ce mode!

Les écrans interlacés

Pour ouvrir un écran interlacé, utilisez ls syntaxe suivante:

SCREEN OPEN 0,320,200,116,LACED [+HIRES] [+LOWRES]

LACED est une fonction qui renvoie la valeur 4.

Important: Dès qu'un écran est ouvert avec le mode Interlacé, tous les autres écrans se mettent sur ce mode. L'interlacement sera seulement véritable pour l'écran qui est déjà ouvert avec LACED. Les lignes de tous les autres écrans seront doublées.

Le mode interlacé est parfait pour afficher des images mais l'animation se déroule à une vitesse qui est deux fois moins rapide. Il n'est pas conseillé d'écrire des jeux en Interlacé!

Dès que le dernier écran interlacé est fermé, vous êtes automatiquement ramené sur le mode normal.

Il se peut que votre moniteur de télévision réagisse mal à de nombreuses permutations rapides entre le mode Interlacé et le mode normal. Il n'est donc pas recommandé d'abuser de ce basculement.

Toutes les opérations normales sont possibles dans des écrans interlacés: Décalage d'écran, Affichage d'écran, etc. Cependant, les plans binaires sont changés lors de l'effacement trame et cet interlacement particulier est interdit lors du calcul de la liste de copper. Ceci s'explique par le fait que l'interlacement est commandé par le logiciel en AMOS.

Si vous avez donc une longue liste de copper (par exemple, 4 écrans, 1 écran interlacé et un arc-en-ciel) et vous devez effectuer un calcul de copper, l'écran interlacé affichera seulement la moitié de l'image pendant le calcul. On ne peut rien faire pour solutionner ce problème, c'est simplement une limitation de l'ensemble du système.

Le mode Hold And Modify (HAM)

Le hardware de l'Amiga vous limite actuellement à un maximum de six plans binaires par écran. Il vous permet d'afficher 64 différentes couleurs à l'écran à la fois. Cependant pour visualiser une image, vous auriez besoin de centaines ou même de milliers de couleurs à l'écran.

Ce fut le problème qui s'est posé à Jay Miner lorsqu'il concevait le système de visualisation d'Amiga. Il l'a résolu en empruntant une astuce bien connue des artistes depuis des siècles.

Si un artiste professionnel devait utiliser chaque couleur possible dans l'élaboration d'un tableau, il serait confronté à une tâche impossible. Il est donc pratique courante de mélanger la teinte exacte le moment venu et de sélectionner cette dernière à partir d'une petite palette de couleurs de base. De cette façon, vous pouvez obtenir des millions de teintes sans avoir à "porter des chargements entiers de peinture".

La même technique peut également s'appliquer à l'écran de l'ordinateur. Au lieu de spécifier chaque couleur, vous pouvez prendre une couleur existante et la modifier légèrement. Ceci augmente fortement le nombre de couleurs possibles et constitue la

base du mode puissant de l'Amiga **H**old **A**nd **M**odify (Ham).

Chaque code couleur établie sur l'Amiga est créée à partir d'un mélange de trois différentes composantes. Ces derniers déterminent l'intensité relative des couleurs primaires Rouge, Vert et Bleu dans la couleur finale. Vous pouvez utiliser des intensités comprises entre 0 et 15.

Le mode Ham divise les codes couleur de l'Amiga en quatre différents groupes:

- **Registres de couleur de 0 à 15:** Les 16 premières couleurs prennent directement leur code du registre de couleurs. Ces couleurs sont simplement considérées comme celles d'un écran standard de 16 couleurs.
- **Composantes Rouges de 16 à 31:** Toutefois, si un point est affecté d'un numéro de couleur de la palette de 16 à 31, le code couleur prend le code du pixel se trouvant à l'immédiate gauche de ce point. La composante Rouge de cette couleur est alors remplacé par un code compris entre 0 et 15 qui est calculé à partir de la formule:
  Intensité=Index de couleur-16
- **Composantes Verts de 32 à 47:** De la même façon, un numéro de couleur compris entre 32 et 47 prend la teinte courante et change la composante Vert. L'intensité de cette composante prend le code couleur 32.
- **Composantes Bleus de 48 à 63:** Ces numéros de couleur saisissent le code couleur à partir du point se trouvant à la gauche du pixel courant et chargent une nouvelle composante bleu à partir du numéro de couleur de la manière suivante:

  Intensité=Index de couleur-48

Le choix de la couleur d'un point particulier dépend des couleurs de tous les points se trouvant à sa gauche. Ceci vous permet de créer de doux dégradés de couleurs qui sont parfaits pour les tons chair. Toutefois, vous ne pouvez pas choisir la couleur de chaque point individuellement. En fait, trois pixels au plus sont nécessaires pour changer de couleurs.

Lorsque l'Amiga fut sorti, le mode Ham était d'abord considéré un peu comme une nouveauté. Aujourd'hui, avec la venue des progiciels graphiques comme Imagen Paint, les choses sont très différentes.

AMOS vous permet d'effectuer toute la gamme d'opérations graphiques et de traitement de texte directement à l'écran Ham. **Le fichier EXEMPLE 10.1** vous donne un exemple simple sur la façon dont vous pouvez produire un écran total avec simplement quelques lignes de code Basic.

Un autre point à considérer: vous pouvez manipuler les écrans Ham en utilisant les commandes normales SCREEN DISPLAY et SCREEN OFFSET. Voici quelques lignes directives simples pour vous aider:

- Le premier point de chaque ligne horizontale doit être affecté du numéro de couleur compris entre 0 et 15. Ce numéro fera office de couleur de départ pour toutes les teintes de la ligne courante.
- N'essayez pas de faire défiler vos écrans Ham horizontalement car vous obtiendriez des franges colorées sur le cadre de votre image. Ces franges sont produites par les changements des couleurs de départ de chaque ligne. Il n'existe aucune restriction quant au défilement vertical.

- Les effets de franges peuvent également produits par la commande SCREEN COPY. Afin d'éviter ce genre d'effets, vérifiez que l'encadrement de votre zone est tracé en utilisant une couleur de la palette de 0 à 15. Vos écrans Ham seront retracés à leur nouvelle position dans leurs couleurs d'origine.

Charger un écran

LOAD IFF *(Charger un écran IFF à partir du disque)*

LOAD IFF "nomfichier"[,écran]

La commande LOAD IFF charge une image de format IFF à partir du disque. Le format IFF est maintenant supporté par la plupart des progiciels graphiques de l'Amiga; vous ne devriez donc pas avoir de difficultés à charger votre propre illustration directement dans l'AMOS Basic.

écran indique le numéro de l'écran qui va être chargé de votre image. Cet écran sera ouvert automatiquement pour votre utilisation. S'il existe déjà, tout ce qui s'y trouve sera complètement effacé.

Pour charger l'image dans l'écran actuel, omettez le paramètre écran. Exemple:

```
Load Iff"AMOS_DATA:IFF/AMOSPIC.IFF",1
```

Sauvegarder un écran

SAVE IFF *(Sauvegarder un écran IFF)*

SAVE IFF "nomfichier"[,compression]

SAVE IFF sauvegarde l'écran courant en tant que fichier d'une image IFF sur disquette. compression est un drapeau vous permettant de choisir si votre fichier sera **compacté** avant d'être sauvegardé. Le code "un" spécifie que le système standard de compression de fichier doit être employé et le code "zéro" sauvegarde l'image comme elle se présente. Les écrans AMOS sont compactés par défaut.

La commande SAVE IFF annexe automatiquement une petite portion IFF à votre fichier image. Elle sauvegarde toutes les manipulations de l'écran effectuées par les commandes SCREEN DISPLAY, SCREEN OFFSET et SCREEN HIDE/SHOW. Lorsque vous chargez ce fichier dans AMOS Basic, il sera renvoyé dans son même état initial. Ces données supplémentaires IFF seront complètement ignorées par les progiciels graphiques externes tels que DPaint 3.

Notez qu'il n'est pas possible de sauvegarder des écrans en dual playfield ou à double buffer avec cette commande.

Déplacer un écran

SCREEN DISPLAY *(Positionner un écran)*

SCREEN DISPLAY n [,x,y,w,h]

Une fois que vous avez défini votre écran avec SCREEN OPEN, il vous faudra la positionner sur votre TV. A la différence de la plupart des autres ordinateurs, l'Amiga est capable d'afficher une image où vous le souhaitez à l'écran de télévision. Vous pouvez facilement exploiter cette possibilité pour produire des effets de rebond sensationnels. Avec AMOS Basic, il est même possible d'obtenir des animations sous interruptions. (voir AMAL).

Une autre possibilité est de recouvrir plusieurs écrans l'un sur l'autre. Ceci vous permet de créer un écran à partir d'une combinaison de différents modes graphiques.

n indique le numéro de l'écran devant être positionné. x et y spécifient la position de l'écran en coordonnées machine.

L'abscisse d'un écran peut être comprise entre 0 et 448 et elle est automatiquement arrondie au multiple inférieur de 16 pixels. Seules les positions comprises entre 112 et 448 sont en fait visibles à l'écran de télévision; nous vous conseillons fortement d'éviter de prendre une abscisse inférieure à 112.

L'ordonnée de votre écran peut être comprise entre 0 et 312. La zone visible est fonction de la taille de votre écran de télévision ou de votre moniteur mais vous trouverez sans doute que les coordonnées comprises entre 30 et 300 conviennent à la plupart des systèmes.

A la date de la rédaction de ce manuel, il paraît y avoir une petite imperfection dans le mode HAM de l'Amiga. Les images en ce mode ne peuvent pas être affichées avec une ordonnée exacte de valeur 256. Il vous faut donc forcer vos coordonnées à des valeurs intermédiaires telles que 255 ou 257. Nous ne savons pas si c'est une erreur de programmation ou une erreur machine mais celle-ci n'imposera aucune restriction sur l'utilisation de ce mode en aucune manière.

w représente la largeur de votre écran en pixels. Si celle-ci est différente de la configuration initiale, seule une portion de votre image sera exposée, depuis l'angle supérieur gauche de l'écran. Comme pour l'abscisse, la largeur de l'écran sera arrondie au multiple inférieur le plus proche de 16 pixels.

De la même façon, *h* détermine la hauteur apparente de l'écran. Le changement de cette valeur réduira la profondeur de votre image.

La commande SCREEN OPEN sélectionnera automatiquement la position de l'écran en utilisant les paramètres standards du fichier de configuration AMOS. Si un écran est plus large que la TV, alors AMOS le configure l'écran de façon à permettre une lecture hors écran.

La commande SCREEN DISPLAY vous offre une façon simple de modifier les valeurs implicites. N'importe quel des paramètres *x,y,h* et *w* peuvent être omis si besoin. Les valeurs inutilisées prendront automatiquement les valeurs implicites et doivent être séparées par des virgules.

```
Screen Display 0,112,45,, : Rem Positionner l'écran à 112,42
```

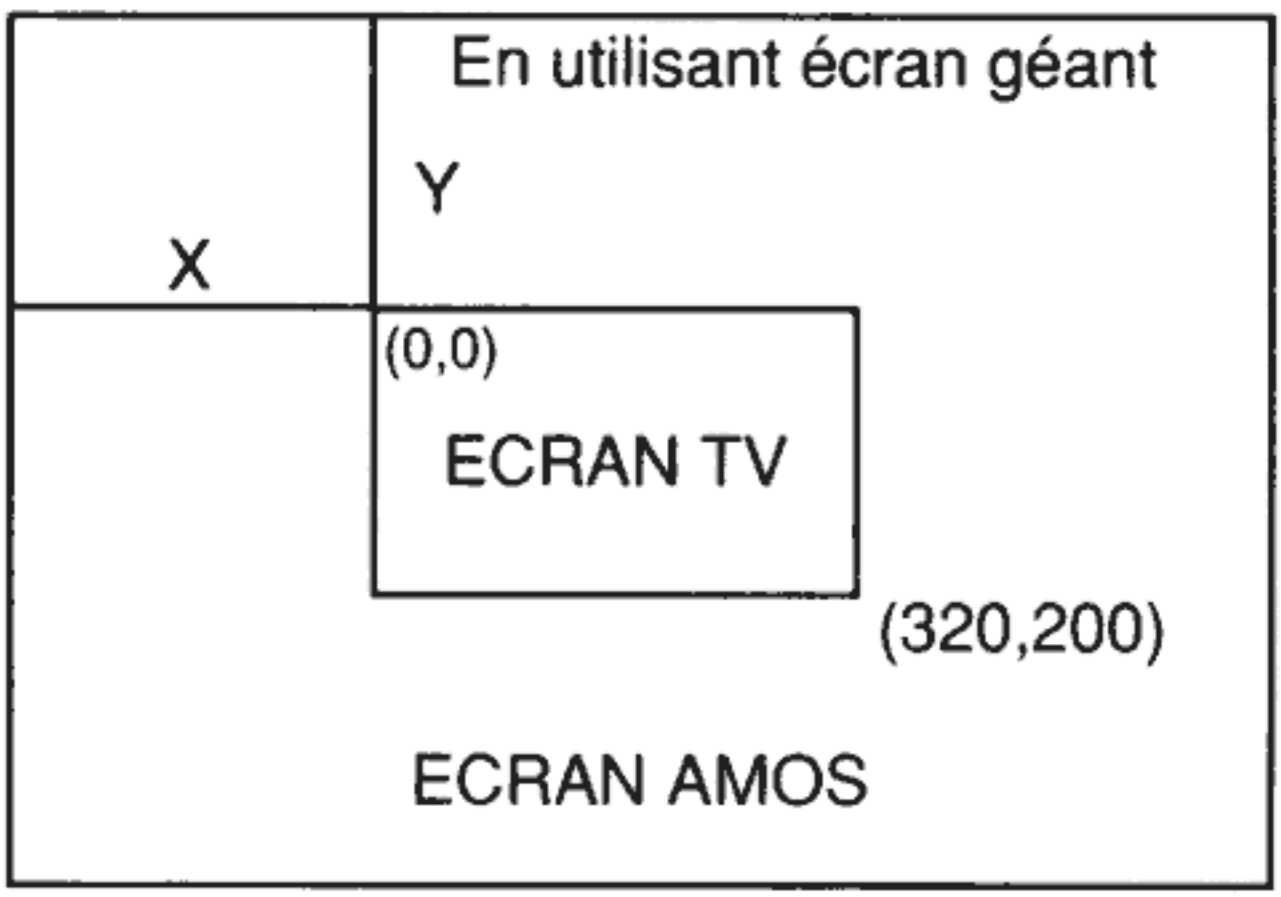
Lorsque vous positionnez vos écrans, assurez-vous que l'écran commence à la gauche de la TV et se termine à sa droite. Ceci est important pour que le hardware de l'Amiga interprète votre écran correctement. En fait, il vous faut peut-être faire quelques essais pour obtenir l'effet précis que vous recherchez. Heureusement, la pire chose qui puisse se produire est l'affichage d'une image bizarre à l'écran. L'Amiga ne se bloquera pas si vous commettez une erreur. Voici quelques lignes directrices qui vous aideront sur votre chemin:

- Seul un écran simple peut être visualisé sur chaque ligne horizontal. Toutefois, vous pouvez placer plusieurs écrans l'un sur l'autre sans risque. Tout se passera bien à condition qu'un des écrans seulement soit visible.
- Il y aura toujours une "zone morte" d'une épaisseur de un pixel entre chaque couple d'écrans. Celle-ci est produite par la liste copper et on ne peut pas du tout l'éviter. Vous pouvez voir la zone morte si vous déplacez un sprite entre les écrans. Par exemple, essayez de déplacer la flèche de la souris de la fenêtre de l'éditeur vers la ligne du menu. Vous devriez apercevoir une petite ligne noire au travers de la flèche de la souris, à la frontière entre les deux écrans.

SCREEN OFFSET *(Scrolling machine)*

SCREEN OFFSET n, x, y

L'image de l'Amiga n'est pas simplement limitée aux dimensions visibles de l'écran de votre télévision. Il n'y a absolument rien vous empêchant de produire des images bien plus larges que l'écran réel. Il est évidemment impossible d'afficher de telles images à l'écran dans leur intégralité, mais vous pouvez facilement visualiser une portion de votre image en utilisant la commande SCREEN OFFSET. Etudiez le schéma suivant:



Comme vous pouvez le constater, la zone sélectionnée de l'écran est visualisée au moyen d'une fenêtre positionnée aux coordonnées x et y. Vous pouvez déplacer la fenêtre au travers de la totalité de l'écran en changeant les valeurs des coordonnées. Ceci produit un scrolling machine lisse qui est parfait pour la plupart des jeux. La taille de votre fenêtre est déterminée à partir des dimensions que vous avez définies par un appel précédent à la commande SCREEN DISPLAY.

n est le numéro de l'écran à visualiser. *x,y* mesurent le déplacement depuis l'angle gauche supérieur de l'écran jusqu'au point de départ de votre image. *x* et *y* sont spécifiés en unités de pixel; par conséquent, vous disposez de tous les moyens pour réaliser des scrollings lents ravissants.

Vous pouvez également faire des retraits grâce à cette instruction, ce qui vous permet d'afficher n'importe quelle partie de la mémoire de l'Amiga à l'écran. Pour une démonstration complète de cette comman0de, voir le fichier **Exemple 10.2.**

Les commandes de manipulation de l'écran

SCREEN CLONE *(Clone un écran)*

SCREEN CLONE n

La commande SCREEN CLONE affecte une deuxième version de l'écran courant au numéro d'écran *n*. Ce clone utilise exactement la même zone- mémoire que l'écran original.

L'écran cloné est normalement visualisé à la même place que son père. Toutefois, vous pouvez le manipuler indépendamment en utilisant n'importe quelle des opérations graphiques normales telles que SCREEN DISPLAY et SCREEN OFFSET.

Puisqu'il n'existe qu'une **seule** copie des données graphiques d'origine en mémoire, vous ne pouvez pas accéder à un clone en utilisant une commande SCREEN. Vous obtiendriez un message d'erreur paramètre d'écran interdit. Un autre point à considérer: les séquences de flash de couleur que vous avez configurées sur l'écran d'origine ne seront pas copiées pendant le clonage. Référez-vous au fichier **EXEMPLE 10.3.** Remarquez l'utilisation de la commande WAIT VBL. Celle-ci permet au clone de se repositionner en dehors de l'écran et aux déplacements de continuer à se dérouler très graduellement.

Si vous faites des essais avec la commande SCREEN CLONE, vous trouverez rapidement que le nombre de mouvements que vous pouvez effectuez sans gâcher l'effet est bien limité. Même quelque chose d'insignifiant comme l'ajout d'un calcul supplémentaire à votre routine de déplacement peut introduire un retard inacceptable dans vos animations.

L'écran peut directement être bougé depuis le langage d'animation AMAL. Ce langage est capable d'animer de grands nombres d'écrans rapidement et facilement. Référez-vous au fichier **EXEMPLE 10.4** pour une démonstration. Les animations sont maintenant si rapides que vous devez les ralentir pour les voir!

DUAL PLAYFIELD *(Combiner deux écrans)*

DUALPLAYFIELD écran1, écran2

Le mode de double terrain vous permet d'afficher deux images entières simultanément à la même abscisse et à la même ordonnée. C'est un peu comme si vous aviez tracé chaque image sur du cellophane et les superposez l'une sur l'autre. Chaque image peut être manipulée individuellement. Vous pouvez utiliser ce procédé pour produire un effet progressif parallaxe qui est parfait pour les jeux avec des scrollings tels que Silkworm.

Les deux composants du dual playfield sont considérés comme toute autre image AMOS et peuvent être écrits de la façon habituelle. Ils peuvent même être animés dans AMAL même ou mis en double buffer.

écran1 and *écran2* se rapportent aux écrans qui ont été préalablement définis avec la commande SCREEN OPEN. Seules certaines combinaisons d'écran sont possibles. Les deux écrans **doivent** utiliser la même résolution, car il est interdit d'utiliser des écrans de haute ou de basse résolution en dual playfield.

Voici une liste de possibilités:

Ecran 1	Ecran 2	Notes
No. des couleurs	No. des couleurs	
2	2	
4	2	
4	4	
8	4	Basse résolution seulement
8	8	Basse résolution seulement

Bien que les palettes de couleurs soient prédéfinies, les tailles des deux écrans peuvent être complètement différentes. En créant un écran en arrière plan plus grand que l'écran de premier plan, vous pouvez créer un réel effet de parallaxe.

Les couleurs de ces écrans sont toutes sélectionnées depuis la palette de *écran 1* avec la couleur zéro considérée comme transparente.

Ecran	Indexes de couleurs (de l'écran 1)
1	de 0 à 7
2	de 8 à 15

Au tracé du deuxième écran, AMOS Basic convertit automatiquement votre index de couleur au numéro approprié avant d'utiliser cet écran. Ainsi, INK 2 se servira de la couleur No 9 de la première palette.

Ce procédé de conversion ne s'applique pas aux instructions d'affectation tels que COLOUR ou PALETTE. Il est important que vous vous rappeliez de cette règle lorsque vous changez les définitions des couleurs, sinon vos nouvelles couleurs ne seront pas reflétées à l'écran réel. Nommez toujours l'écran courant *écran 1* avant de changer vos affectations de couleurs.

Screen 1 : Rem où écran 1 est le numéro du premier écran

Il existe deux ou trois points importants que vous devez connaître avant de configurer un écran en dual playfield:

- Le déplacement d'écran pour les deux écrans ne doit jamais être configuré à zéro.
- Si vous configurez un écran en dual playfield et si vous voulez ensuite le positionner au moyen de la commande SCREEN OFFSET, assurez-vous que l'écran double soit le premier et non le second.

DUAL PLAYFIELD est une instruction très puissante. Une démonstration complète vous est présentée dans le fichier **EXEMPLE 10.5.**

DUAL PRIORITY *(Choisir l'ordre des écrans en dual playfield)*

DUAL PRIORITY écran1,écran2

Le premier écran dual playfield est normalement affiché directement sur le second. La commande DUAL PRIORITY vous permet de changer cet ordre pour que écran2 apparaisse devant *écran1.*

Attention! Cette instruction change seulement l'ordre d'affichage. Elle ne produit **aucun** effet sur la structure de l'écran. Le premier écran de la liste doit donc être utilisé pour toutes les affectations de couleurs en sélectionnant la commande SCREEN DISPLAY.

SCREEN *(Définir l'écran courant)*

SCREEN n

La commande SCREEN vous permet de diriger toutes les opérations graphiques et les opérations de traitement de texte à l'écran numéro *n.* Si cet écran est caché ou s'il est positionné en dehors de la zone d'affichage, la commande ne produira aucun effet visible. Toutefois, les graphiques seront toujours tracés en mémoire et ils seront affichés lorsque l'écran est mis en vue.

=SCREEN *(Saisir le numéro de l'écran courant)*

s=SCREEN

Cette commande renvoie le numéro de l'écran en cours d'utilisation. C'est l'écran dont vous vous servez pour tous les opérations de tracé mais il **n**'est **pas** nécessairement visible.

SCREEN TO FRONT *(Déplacer une image au premier plan)*

SCREEN TO FRONT [s]

Cette instruction met l'écran s au premier plan de l'image de télévision. Si vous omettez le paramètre s, l'écran courant sera alors utilisé.

```
Screen Close 0 : Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF",0
Screen Open 1,320,200,16,Lowres
Centre "Ecran 1!"
Wait Key : Screen To Front 0
```

Remarque: Si le système AUTOVIEW a été éteint, il vous faudra appeler la commande VIEW avant que l'effet soit visible à l'écran.

SCREEN TO BACK *(Déplacer une image au fond de l'écran)*

SCREEN TO BACK[n]

La commande SCREEN TO BACK met une image à l'arrière-plan de votre écran. S'il existe une autre image aux mêmes coordonnées, celle-ci sera alors affichée au premier plan de l'écran sélectionné.

SCREEN HIDE *(Cacher temporairement un écran)*

SCREEN HIDE [n]

Cette commande retire complètement un écran sélectionné de la visualisation. Vous pouvez réafficher cet écran en faisant un appel à la commande SCREEN SHOW. Si n est omis, cette instruction cachera l'écran courant.

SCREEN SHOW *(Remettre un écran à son état initial)*

SCREEN SHOW n

La commande SCREEN SHOW renvoie un écran à l'affichage après qu'il ait été caché avec la commande SCREEN HIDE.

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF",1
Screen Hide : Wait Key : Screen Show
```

=SCREEN HEIGHT *(Renvoyer la hauteur de l'écran courant)*

h=SCREEN HEIGHT [n]

Cette commande renvoie la hauteur d'un écran AMOS. Si vous ne précisez pas le paramètre de cette instruction, la hauteur renvoyée sera celle de l'écran courant.

Print Screen Height

=SCREEN WIDTH *(Renvoyer la largeur de l'écran courant)*

w=SCREEN WIDTH [n]

La commande SCREEN WIDTH retrouve la largeur de l'écran courant ou de l'écran numéro n.

Print Screen Width

=SCREEN COLOUR *(Renvoyer le nombre de couleurs)*

c=SCREEN COLOUR

L'instruction SCREEN COLOUR renvoie le nombre maximal de couleurs dans l'écran en cours d'utilisation.

Print Screen Colour

=SCIN *(Renvoyer le numéro d'écran à une position sélectionnée)*

s=SCIN(x,y)

Cette instruction renvoie le numéro de l'écran qui se trouve dessous les coordonnées **machine** *x,y*. Si cet écran n'existe pas, s prendra alors une valeur négative (nulle).
SCIN est normalement utilisée en combinaison avec les fonctions X MOUSE et Y MOUSE pour vérifier si le curseur de la souris est entré dans un écran particulier. Exemple:

Print Scin(X Mouse, Y Mouse)

Définir les couleurs de l'écran

DEFAULT PALETTE *(Charger l'écran d'une palette standard)*

DEFAULT PALETTE c1,c2,c3,,,c6,,,—> jusqu'à 32 couleurs

Cette commande simplifie le procédé d'ouverture de nombreux écrans en permettant l'utilisation de la même palette. Elle définit une liste de couleurs qui seront utilisées pour tous les écrans suivants créés à l'aide de l'instruction SCREEN OPEN. Comme d'habitude, les codes couleur permissibles sont compris entre $000 et $FFF (RVB). Voir également à la commande GET SPRITE PALETTE.

GET PALETTE *(Définir la palette d'un écran)*

GET PALETTE n [,masque]

L'instruction GET PALETTE copie les couleurs de l'écran *n* et les charge dans l'écran courant. Elle peut s'avérer être très utile lors du déplacement des informations d'un écran à l'autre avec la commande SCREEN COPY, puisqu'il est essentiel que les définitions de couleurs les écrans source et de destination soient les mêmes.

Le paramètre optionnel *masque* vous permet de charger une sélection de couleurs. Voir la commande GET SPRITE PALETTE pour plus de précisions sur *masque*. Exemple:

```
Load iff "AMOS_DATA:Iff/AMOSPIC.IFF",0
Screen Open 1,Screen Width, Screen Height, Screen Colour, Lowres
Screen Copy 0,0,0,160,100 To 1,80,80
Centre "<Appuyez sur une touche pour saisir la palette>" : Wait Key
Get Palette 0
Rem Effacer l'écran
```

CLS *(Effacer l'écran)*

CLS efface l'intégralité ou une partie de l'écran courant. Il existe trois variantes possibles de cette commande.

CLS

Efface l'écran courant en le remplissant de la couleur zéro et efface toutes les fenêtres pouvant avoir été définies.

CLS col

Remplit votre écran avec couleur col.

CLS col,x1,y1 to x2,y2

Remplit la zone rectangulaire aux coordonnées *x1,y1,x2,y2* d'une couleur pleine *col. col* peut prendre n'importe quelle valeur, de zéro au nombre de couleurs disponibles.

x1,y,x2,y2 représentent les coordonnées de l'angle supérieur gauche et de l'angle inférieur droit de l'écran devant être effacé par cette commande. Exemple:

```
Cls : Circle 100,98,98 : Cls 1,50,50 To 150,150
```

Manipuler le contenu d'un écran

SCREEN COPY *(Copier des portions d'écran)*

SCREEN COPY scr1 TO scr2

SCREN COPY scr1,x1,y1 ,x2,y2 TO scr2,x3,y3 [,mode]

La commande SCREEN COPY permet de copier de grandes portions d'écran d'un endroit à l'autre à une vitesse incroyable.

scr1 représente l'écran utilisé en tant qu'origine de votre image. Ce peut être un numéro d'écran standard ou le numéro d'un écran physique ou logique crée en utilisant les commandes LOGIC ou PHYSIC.

scr2 sélectionne un écran optionnel de destination dans lequel ces données seront copiées. S'il est omis, la zone sera copiée dans l'écran courant.

z1,y1 et x2,y2 représentent les dimensions d'une zone rectangulaire source, et *x3,y3* les coordonnées de la zone de destination. Il n'existe aucune restriction imposée au choix de ces coordonnées. Les portions de votre image se trouvant en dehors de la zone de l'écran courant seront automatiquement coupées.

Vous trouverez un exemple d'illustration de la commande SCREEN COPY dans le fichier **EXEMPLE 10.6** du dossier du MANUEL.

Le paramètre optionnel mode choisit parmi 255 modes possibles de manipulation d'objets le mode qui sera utilisé pour votre copie. Ces modes déterminent le type de combinaison des zones source et de destination à l'écran. Le mode est configuré en utilisant une configuration binaire dont le format est le suivant:

Mode binaire	**Bit d'origine**	**Bit de destination**
4	0	0
5	0	1
6	1	0
7	1	1

Notez que les quatre derniers bits de cette configuration ne sont pas utilisés par cette instruction et doivent toujours être forcés à zéro.

Chaque bit en *mode binaire* représente une seule combinaison de bits dans les zones source et de destination. Si un mode binaire est configuré à un, alors le bit correspondant à l'écran est également entré avec un 1, sinon le résultat sera 0.

Pour sélectionner le mode correct de tracé, il vous suffit de trouver les combinaisons donnant pour résultat une valeur de un et de configurer les bits appropriés dans le paramètre du mode en conséquence.

Supposez que vous vouliez seulement configurer un bit à l'écran si les bits source et de destination sont identiques. Vous cherchez donc dans le tableau les points correspondants à votre spécification. La valeur suivante du mode serait ainsi produite:

%10010000

Si vous n'êtes pas familier avec la numération binaire, il se peut que vous trouviez cette commande un peu obscure. Au lieu de vous ennuyer à mort avec une explication de programme en binaire, nous avons préféré vous donner une liste détaillée des spécifications les plus courantes avec les mode points correspondants.

Mode	Effet	Configuration binaire
REMPLACER	Remplace la zone de destination par une copie directe de l'image source (défaut).	%11000000
INVERSER	Remplace l'image de destination par une copie inversée de l'image source.	%00110000
ET	Combine la source et la destination avec un ET logique.	%10000000
OU	Combine par une porte OU l'origine et l'image de destination.	%11100000
OU EXCLUSIF	Combine la zone source et la zone de destination avec un OU exclusif.	%01100000

Les utilisateurs ayant l'esprit technique devraient noter que la commande SCREEN COPY combine la source et la destination en utilisant les zones de manipulation d'objets B et C et non la zone A qui n'est pas du tout utilisée par le système.

Faire défiler l'écran

DEF SCROLL *(Définir une zone de scrolling)*

DEF SCROLL n,x1,y1 à x2,y2,dx,dy

La commande DEF SCROLL vous permet de définir 16 différentes zones de scrolling. Chacune de ces zones peuvent être associées à un scrolling spécifique qui est déterminé par les variables *dx* et *dy*.

n représente le numéro de la zone pouvant être compris entre 1 et 16. *x1,y1* se rapportent aux coordonnées de l'angle supérieur gauche de la zone à défiler et *x2,y2* au point diagonalement opposé.

dx signifie le nombre de pixels compris dans le report de la zone sur la droite à chaque opération. Les numéros négatifs indiquent que le scrolling se fera de droite à gauche et les numéros positifs de gauche à droite.

De la même façon, *dh* représente le nombre de points compris dans l'ascension ou la descente de la zone pendant le scrolling. Dans ce cas, les valeurs négatives de *dy* sont utilisées pour indiquer un mouvement ascendant et les valeurs positives pour signaler un mouvement descendant.

SCROLL *(Faire défiler l'écran)*

SCROLL n

La commande SCROLL fait défiler l'écran en utilisant les positions que vous lui avez

spécifiée à l'aide de l'instruction DEF SCROLL. *n* se rapporte au nombre de la zone que vous souhaitez faire défiler.

```
Load Iff "AMOS_DATA:IFF/Frog_Leap.IFF",2
Def Scroll 1,0,0 to 320,200,1,0
Do
   Scroll 1
Loop
```

Vous pouvez trouver de plus longs exemples dans les fichiers **EXEMPLE 10.7** et **EXEMPLE 10.8.** Ces exemples chargent une image de la disquette système AMOS et la font tourner autour de l'écran. La variable S contient le nombre de points de déplacement de l'image lors de chaque exécution de la commande. Plus la valeur de S est grande, plus le scrolling est rapide et saccadé. Notez que l'utilisation de la bascule d'écran améliore la qualité du déplacement.

Bascule d'écran

Afin de produire les effets de déplacements sans saccade comme dans un jeu de commerce, il est nécessaire de terminer toutes les opérations de traçé en moins d'un cinquantième de seconde. Ceci représente un vrai défi pour l'ordinateur le plus rapide et il est souvent impossible de le relever même sur l'Amiga. Si l'animation est complexe, vos graphismes auront donc tendance à scintiller d'une façon agaçante lors de leur tracé.

Heureusement, il existe une solution à la portée de la main dont l'exploitation s'est avérée être une réussite pour la plupart de jeux d'arcade modernes. La technique de bascule d'écran peut facilement produire une animation d'écran exempte de scintillement en utilisant juste une fraction de la puissance de calcul de l'Amiga.

L'idée de base est extrêmement simple. Au lieu de construire vos images sur l'écran réel, vous effectuez toutes vos opérations de tracé sur un écran logique indépendant qui est totalement invisible. Cet écran se distingue de l'écran physique qui est actuellement affiché sur votre télévision. Une fois que les graphiques ont été terminés, vous pouvez alors permuter les écrans logiques et physiques pour produire une transition graduelle entre les deux images. L'ancien écran physique devient alors le nouvel écran logique et il est utilisé pour élaborer l'image suivante de votre séquence.

Au premier coup d'oeil, ce procédé semble assez compliqué, mais il est exécuté automatiquement par la commande DOUBLE BUFFER d'AMOS Basic. Celle-ci force toutes les opérations de tracé à s'effectuer directement sur l'écran logique sans affecter l'écran courant. Il vous suffit seulement de rendre vos opérations de tracé synchrones avec les bascules d'écran dans votre programme. Vous pouvez réaliser cet effet en utilisant l'instruction SCREEN SWAP.

SCREEN SWAP *(Permuter les écrans physiques et logiques)*

SCREEN SWAP [n]

La commande SCREEN SWAP permute les écrans physiques et logiques. Elle vous

permet d'intervertir instantanément l'écran physique entre les deux écrans.

Si vous utilisez le DOUBLE BUFFER, ces écrans auront déjà été créés pour vous. Toutefois, il vous faudra éteindre le système automatique de bascule d'écran à l'aide de BOB UPDATE OFF, sinon les écrans seront permutés 50 fois par seconde et pertuberont vos propres opérations de tracé. Il est également nécessaire de désactiver la fonction autoback à l'aide de AUTOBACK OFF. Cette fonction copie normalement vos opérations graphiques sur les écrans physiques et logiques. Elle est utile lorsque vous souhaitez combiner de simples graphiques aux bobs actifs mais elle détruit complètement l'effet des opérations de bascule de l'écran.

Pour illustrer la puissance de cette commande, jetez un coup d'oeil aux programmes **EXEMPLE 10.9** et **EXEMPLE 10.10.**

Le programme **EXAMPLE 10.9** déplace un triangle au travers de l'écran sans utiliser une seule forme de bascule d'écran. Tandis que le triangle avance, il produit un scintillement agaçant et intense. Ajoutons maintenant une petite bascule d'écran à ce programme. Nous allons dessiner le triangle sur un écran logique invisible et le mettre en vue une fois tracé. Vous pouvez trouver ce nouveau programme dans **EXEMPLE 10.10.**

Le programme **EXAMPLE 10.10** trace chaque nouveau triangle à l'écran sans affecter l'affichage courant. L'instruction SCREEN SWAP permute alors les écrans logiques et physiques afin que la version terminée du triangle apparaisse immédiatement à l'écran sans un moindre scintillement. L'ancien triangle est maintenant effacé de l'écran logique et il est retracé à la position suivante. Tandis que le programme se déroule, le triangle avancera lentement d'un bord à l'autre de l'écran.
Remarquez que nous avons fait exprès d'exagérer le scintillement dans **EXEMPLE 10.9** pour illustrer la technique de bascule d'écran. Il serait en fait assez facile de réduire considérablement cet effet même sans utiliser l'instruction SCREEN SWAP.

=LOGBASE (Renvoyer l'adresse d'une portion de l'écran logique)

adresse=LOGBASE(plan)

La fonction LOGBASE est destinée aux programmeurs experts qui souhaitent accéder directement à la mémoire de l'écran d'Amiga.

plan se rapporte au six plans binaires possibles composant l'écran courant. Après avoir appelé LOGBASE, adresse contiendra soit l'adresse du plan binaire demandé, soit la valeur zéro si elle n'existe pas.

=PHYBASE *(Renvoyer l'adresse de l'écran courant)*

adresse=PHYSBASE(plan)

La commande PHYSBASE renvoie l'adresse en mémoire du plan binaire numéro plan de l'écran courant. Si ce plan n'existe pas, cette fonction renvoie alors une valeur de zéro.

Loke Phybase(0),0 : Rem Trace un extra plat directement à l'écran

=PHYSIC *(Renvoyer l'identificateur de l'écran physique)*

=PHYSIC
=PHYSIC(s)

La fonction PHYSIC renvoie le numéro d'identification de l'écran physique courant. Ce numéro vous permet d'accéder directement à l'image physique qui est à présent affichée par le système de double tampon.

Vous pouvez remplacer le résultat de cette fonction par le numéro d'écran dans les commandes ZOOM, APPEAR et SCREEN COPY.

s est le numéro de l'écran AMOS. S'il est omis, l'écran actuel sera alors utilisé. Ne la confondez pas avec la fonction LOGBASE.

=LOGIC *(Renvoyer l'identificateur de l'écran logique)*

LOGIC
=LOGIC(s)

Cette fonction renvoie le numéro d'identification d'un écran logique. Elle peut être utilisée en combinaison avec les commandes SCREEN COPY, APPEAR et ZOOM pour changer votre image hors-écran, sans affecter l'écran courant.

Cadrage de l'écran

Comme la plupart des ordinateurs individuels, l'AMIGA utilise un écran à carte mémoire. Ce terme technique s'applique à un concept qui vous est sans doute déjà familier. En voici une simple explication: un écran à carte mémoire est un écran qui utilise un équipement spécial pour convertir une image conservée en mémoire en un signal pouvant être affiché à votre écran de télévision. Lorsqu'AMOS Basic accède à l'écran, il le fait au moyen de cette mémoire d'écran.

L'écran est raffraîchi par la machine chaque 50ème de seconde. Une fois qu'une image a été tracée, le faisceau d'électrons s'arrête et revient à sa position en haut et à gauche de l'écran. Ce procédé est appelé signal fin de balayage. AMOS Basic effectue simultanément un certain nombre de tâches importantes telles que le déplacement des sprites et la bascule de l'adresse de l'écran physique si celle-ci a changé. Les actions des instructions telles que ANIM ou SCREEN SWAP seront seulement entièrement exécutées lorsque l'écran est retracé.

Puisqu'un 50ème de seconde représente bien un long moment pour l'AMOS Basic, ceci peut amener un sérieux manque de coordination entre votre programme et l'image, ce qui peut être particulièrement remarqué dans les boucles étroites des programmes. La meilleure façon d'éviter cette difficulté est d'attendre que l'écran soit raffraîchi avant d'exécuter la commande Basic suivante.

WAIT VBL *(Attendre un signal de fin de balayage)*

WAIT VBL

L'instruction WAIT VBL arrête l'AMIGA jusqu'au signal de fin de balayage. Elle est couramment utilisée après une instruction PUT BOB ou SCREEN SWAP.

Les effets spéciaux

APPEAR *(Apparition graduelle d'une image)*

APPEAR source TO destination,effect [,pixels]

La commande APPEAR vous permet de produire de jolies apparitions graduelles entre les écrans source et *de destination.* Source et *destination* sont simplement les numéros d'écrans que vous avez préalablement ouverts en utilisant SCREEN OPEN. Vous pouvez également les remplacer par les fonctions LOGIC et PHYSIC à ces positions si besoin.

effet détermine le type d'apparition d'une image produit par cette instruction. La taille de ce paramètre est comprise entre 1 et le nombre de pixels de votre écran courant.

pixels spécifie le nombre de points qui doivent être affectés. Cette valeur est normalement initialisée à la zone d'écran TOTAL, mais vous pouvez la réduire pour créer une apparition graduelle sur une seule partie de l'écran. Tous les écrans sont tracées dans un ordre bien déterminé de haut en bas de l'écran.
L'apparence de vos apparitions graduelles vont naturellement varier en fonction du mode d'écran que vous utilisez. Le programme **EXEMPLE 10.11** vous permette de mettre en pratique les diverses possibilités.

FADE *(Mélanger une ou plusieurs couleurs à de nouveaux codes couleur)*

FADE speed [,couleur liste]
FADE speed TO screen [,masque]

La commande FADE vous permet de changer graduellement toute la palette en passant d'un ensemble de couleurs à un autre. Cette commande peut être utilisée pour produire des effets d'apparition graduelle et donner un look professionnel à vos écrans d'accueil.

La version classique de cette instruction saisit la palette courante et dissout lentement les couleurs de l'écran. Chaque code couleur est réduite l'une après l'autre de un jusqu'à zéro. Exemple:

```
Fade 15:Wait225
```

speed est le nombre d'impulsions de suppression images devant avoir lieu avant que le nouveau changement de couleur se fasse.
Puisque les effets d'apparition graduelle sont produits en utilisant des interruptions, il est préférable d'attendre que l'opération soit entièrement terminée avant de passer à l'instruction Basic suivante. Le temps d'ATTENTE de l'apparition graduelle peut être calculée par la formule suivante:

valeur d'attente = vitesse d'apparition * 15

La commande FADE peut être étendue pour produire une nouvelle palette directement à partir d'une liste de codes couleur.

```
Fade 15,$100,$200,$300
```

N'importe quel numéro de couleurs permis dans le mode graphique courant peut être spécifié dans cette instruction. Comme la plupart des commandes AMOS, il est possible d'omettre complètement les paramètres sélectionnés. Ces couleurs seront non affectées par la commande FADE.

```
Fade 15,,$100,$800,$F00
```

La forme la plus puissante de FADE transforme graduellement les couleurs de l'écran courant en une palette prise dans l'écran existant.

FADE speed TO s[,masque]

Les présentes couleurs sont lentement converties en la palette de l'écran *s*. Il est également possible de charger la palette de la banque sprites en utilisant la même technique. Il vous suffit de donner une valeur négative au numéro d'écran *s*.
masque est une configuration binaire spécifiant les couleurs devant être chargées. Chaque couleur est associée à un seul bit de cette configuration numérotée de 0 à 31. Si un bit est forcé à 1, alors la couleur appropriée sera changée. Référez-vous au fichier **EXEMPLE 10.12.**

FLASH *(Définir la séquence de couleurs clignotantes)*

Cette commande vous donne la possibilité de changer périodiquement la couleur affectée à un indexe quelconque de couleurs. Elle procède à ce changement en intégrant une interruption similaire à celle utilisée par les instructions de sprite et de musique. La syntaxe de l'instruction flash est:

FLASH index,"(couleur,attente)(couleur,attente)(couleur,attente)..."

Index est le numéro de la couleur qui doit être animée. Attente est défini en unités de 50ème de seconde.

Couleur est mémorisé dans le format standard RVB (Voir COLOUR pour plus de précisions). FLASH a pour tâche de prendre chaque nouvelle couleur de la liste, l'une après l'autre, et de les charger dans l'index pendant une durée spécifiée par attente. Lorsqu'elle arrive à la fin de la liste, elle répète l'entière séquence de couleurs depuis le début. Notez que vous êtes seulement autorisé à utiliser au plus 16 changements de couleur dans une intruction FLASH quelconque. Voici un petit exemple:

```
Flash 1,"(007,10)(000,10)"
```

FLASH utilise le code couleur 1 alternativement entre le bleu et le noir chaque 10/50

(1/5ème) de seconde. Etudions maintenant quelque chose de plus complexe:

Flash 0,2"(111,2)(333,2)(555,2)(777,4)(333,4)"

Si ceci vous donne un mal de tête, vous apprendrez avec joie que vous pouvez désactiver le clignotement en utilisant l'instruction:

FLASH OFF

Notez également qu'au démarrage, le code couleur 3 est automatiquement affecté à une séquence flash qui sera utilisée par le curseur. Il est bon de désactiver cette fonction avant de charger des images du disque.

SHIFT UP *(Rotation de couleurs)*

SHIFT UP attente,premier,dernier,indicateur

La commande SHIFT UP fait tourner les codes contenus dans les registres de couleur du *premier* au *dernier*. La *première* couleur de la liste est copiée sur la seconde, la seconde sur la troisième et ainsi de suite jusqu'à ce que la *dernière* couleur de la série soit saisie.

Chaque écran AMOS peut avoir sa propre et unique palette d'animations de couleurs. Les changements de couleurs peuvent être utilisés pour créer des séquences captivantes d'hyperespace similaires à celles rencontrées dans Captain Blood et Elite. Puisque ces animations sont réalisées en utilisant des interruptions, elles peuvent être exécutées lors du déroulement de votre programme sans aucunement l'affecter.

attente est l'intervalle de temps compris entre chaque étape de rotation, mesuré en 50ème de seconde.

indicateur commande le type de rotation. S'il est forcé à un, le dernier index de couleur de la liste sera copié sur le premier et le premier sur le dernier. Ainsi, les couleurs tourneront de façon continue à l'écran. Lorsque *indicateur* est forcé à zéro, le contenu des *premier* et *dernier* indexes seront jetés et la zone comprise entre le *premier* et le *dernier* sera graduellement remplacée par une copie de la première couleur de la liste. Par exemple:

Shift Up 100,1,15,1

Shift Up 10,1,15,0

Ces changements de couleur peuvent être désactivés au moyen de la commande SHIFT OFF.

SHIFT DOWN *(Faire tourner une liste de codes couleur)*

SHIFT DOWN attente,premier,dernier,indicateur

SHIFT DOWN est similaire à la commande précédente SHIFT UP, à l'exception qu'elle fait tourner les couleurs dans la direction opposée. Ainsi, la deuxième couleur sera copiée sur la première, la troisième sur la seconde et ainsi de suite.

premier et *dernier* saisissent une liste d'indexes de couleur à faire défiler. attente spécifie un intervalle entre chaque changement de couleur en unités d'un 50ème de seconde.

indicateur détermine le type de rotation. Une valeur de un donne lieu à un cycle continu de couleurs et un zéro change les couleurs sans sauvegarder les contenus d'origine des indexes *premier* et *dernier.* Après un cycle complet, toutes les couleurs comprises entre les indexes premier et *dernier* contiendront une copie de la couleur contenue dans le *dernier* index. Voir également FLASH, PALETTE et COLOUR.

SHIFT OFF *(Arrêter tous les cycles de couleur pour l'écran courant)*

SHIFT OFF

SHIFT OFF met immédiatement fin à toutes les rotations de couleurs générées par les instructions SHIFT UP ou SHIFT DOWN.

SET RAINBOW *(Définir un effet d'arc-en-ciel)*

SET RAINBOW n,couleur,longueur,r$,g$,b$

La commande SET RAINBOW définit un joli effet d'arc-en-ciel qui peut être affiché par la suite au moyen de la commande RAINBOW. Celle-ci s'exécute en changeant la teinte d'une couleur selon une série de règles simples.

n représente le numéro de votre arc-en-ciel. Vous pouvez choisir sa valeur entre 0 et 3. *couleur* est un index de couleurs qui sera changé par l'instruction. Cette couleur peut être affectée d'une différente valeur pour chaque ligne d'écran horizontale (ou *ligne de balayage*). Notez que seules les couleurs comprises entre 0 et 15 peuvent être manipulées au moyen de ce système.

longueur définit la taille du tableau utilisé pour ranger vos couleurs. Une entrée est prévue dans le tableau pour chaque code couleur affiché à l'écran. La taille de ce tableau peut être comprise entre 16 et 65500. Si *longueur* est inférieure à la hauteur réelle de votre arc-en-ciel, alors le motif de couleur sera répété plusieurs fois à l'écran.

Les chaînes de commande *r$,g$,b$* changent progressivement les intensités des composants du rouge, du vert et du bleu pour former votre couleur finale. Ces codes sont chargés dans un tableau de couleurs spécial. Chaque couleur du tableau détermine l'apparence d'une seule ligne de balayage horizontale à l'écran.

Au début de l'arc-en-ciel, tous les composants de votre couleur sont chargés de la valeur zéro. Cette valeur changera selon l'information contenue dans le tableau de couleurs.

Les chaînes de commande peuvent être omises si besoin, mais il vous faudra toujours placer les guillemets et les virgules à leurs positions appropriées.

Chaque chaîne peut contenir une liste complète de commandes. Celles-ci seront

répétées de façon continue pour produire la forme finale d'un arc-en-ciel. La syntaxe est la suivante:

(n,pas,nombre)

n détermine le nombre de lignes devant être affectées à un code couleur spécifique dans l'arc-en-ciel. L'accroissement de ce nombre changera la hauteur de chaque ligne de l'arc-en-ciel.

pas contient une valeur devant être ajoutée au composant. Cette valeur sera utilisée pour produire la couleur de la ligne suivante à l'écran. Un *pas* positif augmentera l'intensité du composant de la couleur et une valeur négative la réduira. Si un certain composant dépasse 15 qui est la valeur la plus élevée, une nouvelle valeur sera calculée à partir de la formule suivante:

nouveau composant=ancien composant Mod 15

nombre est le nombre de répétitions de l'opération courante. La meilleure façon de démontrer cette commande est de l'illustrer d'un exemple. Tapez les lignes suivantes:

```
Set Rainbow 0,1,64,"(8,2,8)","",""
Rainbow 0,56,1,255 : Rem Affiche l'arc-en-ciel
Wait key
```

Ceci crée un nouvel arc-en-ciel de numéro zéro utilisant l'index de couleur un. Comme vous pouvez le constater, SET RAINBOW définit seulement votre arc-en-ciel. Afin de l'afficher à l'écran, il vous faut utiliser la commande RAINBOW (voir ci-dessous).

L'effet arc-en-ciel affecte d'abord la valeur zéro à votre couleur. Le composant rouge est alors automatiquement incrémenté de deux unités toutes les quatre lignes de balayage. Ainsi, la valeur de la couleur zéro change progressivement de $000 à $E00. Lorsque le composant dépasse 15, son reste est calculé et la couleur est renvoyée au point de départ (zéro). Le schéma sera alors répété plus bas à l'écran.

En définissant un motif individuel pour chaque composant de rouge, de vert et de bleu de votre couleur, vous pouvez facilement produire de remarquables motifs à l'écran. Puisque chaque arc-en-ciel utilise seulement un seul index de couleurs, rien ne vous empêche de créer des effets identiques en utilisant simplement deux écrans couleur. Ceux-ci sont parfaits pour les décors d'un jeu d'arcade puisqu'ils consomment très peu de mémoire. Exemple:

```
Screen Open 0,320,256,2,Lowres
Set Rainbow 0,1,128,"(8,1,8)","(8,1,8)",""
Rainbow 0,1,30,128
Colour 1,0 : Curs Off : Cls 1 : Flash Off
Locate 0,2 : Centre "Amos Basic" : Wait Key.
```

Pour étudier une autre démonstration des superbes effets pouvant être obtenus avec cette instruction, chargez le fichier **EXAMPLE 10.13.**

Vous pouvez également animer les arcs-en-ciel en utilisant un puissant système

d'interruption. Référez-vous au chapitre AMAL pour plus de précisions.

RAINBOW *(Créer un effet d'arc-en-ciel)*

RAINBOW n,base,y,h

La commande RAINBOW affiche un numéro d'arc-en-ciel *n* à l'écran. Si AUTOVIEW est positionné sur OFF, l'arc-en-ciel apparaîtra seulement à l'appel de la commande VIEW.

base est le déplacement de la première couleur du tableau que vous avez créé avec SET RAINBOW. Le changement de cette valeur bouclera le cycle de l'arc-en-ciel à l'écran.

y détermine la position verticale de l'arc-en-ciel exprimée en coordonnées machine. La valeur minimale de cette coordonnée est 40. Si vous essayez de prendre une coordonnée inférieure à ce point, l'arc- en-ciel s'affichera à partir de la ligne 40.

h définit la hauteur de votre arc-en-ciel en lignes de balayage.

Les arcs-en-ciel sont entièrement compatibles avec le système AMOS, les bobs et sprites compris. Toutefois, n'essayez pas de former le dégradé d'une couleur qui est en cours de modification en utilisant les instructions FLASH ou SHIFT, puisque ceci entraînera des effets imprévisibles à l'écran.

Notez qu'un seul effet d'arc-en-ciel peut être affiché sur une certaine ligne de balayage, même si différentes couleurs sont utilisées à l'écran.

Normalement, l'arc-en-ciel dont la position est la plus haute à l'écran est d'abord affichée. Mais si plusieurs arcs-en-ciel commencent depuis la même ligne de balayage, alors l'arc-en-ciel dont le numéro d'identification est le plus petit sera tracé devant les autres.

=RAIN *(Changer la couleur d'une ligne individuelle de l'arc- en-ciel)*

RAIN(n,ligne)=c
c=RAIN(n,ligne)

C'est la commande la plus puissante de toutes les commandes de création d'arc-en-ciel puisqu'elle vous permet de changer le code couleur d'une ligne particulière de l'arc-en-ciel.

n est le numéro d'arc-en-ciel que vous souhaitez accéder. ligne est la ligne particulière de balayage devant être changée. Exemple:

```
Curs Off:Centre "Un Arc-en-ciel AMOS!"
Rem définir un arc-en-ciel à partir de valeurs fictives.
Set Rainbow 1,1,4097,"","",""
For Y=0 To 4095
   Rem Charger un arc-en-ciel d'un code couleur compris entre 0 et 4095
   Rain(1,Y)=Y
Next Y
For C=0 To 4095-255
   Rem afficher les 255 lignes de l'arc-en-ciel depuis la ligne 40
```

```
    Rainbow 1,C,40,255
  Next C
  Wait Key
```

Ceci fait défiler graduellement l'intégralité de la palette en commençant par le code couleur un.

ZOOM *(Agrandir une portion de l'écran)*

ZOOM source,x1,y1,x2,y2 TO dest, x3,y3,x4,y4

ZOOM est une instruction simple vous permettant de changer la taille de toute zone rectangulaire de l'écran.

source est le numéro d'un écran à partir duquel votre image est saisie. Vous pouvez également utiliser la fonction LOGIC pour saisir votre image de l'écran logique approprié. Cette zone rectangulaire devant être affectée par cette instruction est entrée en utilisant les coordonnées *x1,y1,x2,y2. x1,y1* déterminent la position dans l'angle supérieure gauche de cette zone et *x2,y2* définissent les coordonnées de l'angle diagonalement opposé. dest contient l'écran de destination de votre image. Comme la source, il peut être soit un numéro d'écran soit un écran logique spécifié en utilisant LOGIC.

Les dimensions de cet écran sont déterminées à partir des coordonnées *x3,y3 et x4,y4.* Ces dernières représentent les dimensions du rectangle dans lequel la portion d'écran sera comprimée.

L'effet de cette instruction est fonction de la taille relative des rectangles source et de destination. L'image source est automatiquement remodelée pour entrer parfaitement dans les rectangles de destination. Ainsi, la même instruction peut être utilisée pour réduire ou agrandir vos images à votre convenance. Voici un exemple:

```
  F$=Fsel$("*.*","","Charger écran") : If F$="" then Direct
  Load Iff F$,0 : Screen Open 1,320,256,Screen Colours,Lowres
  Flash off : Get Palette(0)
  Screen Display 1,,,,256 : View : Limit Mouse
  Repeat
    Zoom 0,0,70,320,175 To 1,0,0,X Screen (X Mouse)+1,Y Screen (Y
MOUSE)+1
  Until Mouse Key
```

Vous pouvez trouver une autre démonstration dans le fichier **EXAMPLE 10.14.**

Changer la liste copper

Le coprocesseur de l'Amiga (copper) vous permet d'exercer un parfait contrôle sur l'apparence de chaque ligne de votre écran. Ce copper est un processeur indépendant doté de sa propre mémoire interne et de son unique jeu d'instructions. En programmant le copper, il est possible de produire aisément une gamme impressionnante d'effets à

l'écran. Normalement, le copper est géré automatiquement par le système AMOS. Chacun des effets possibles du copper peuvent être produits directement depuis AMOS Basic sans avoir à se livrer à une programmation compliquée. En pratique, ces instructions seront bien suffisantes pour la plupart des applications.

Evidemment, personne ne peut penser à tout. Les programmeurs experts peuvent souhaiter avoir un accès direct au copper pour créer leurs propres modes d'écran spéciaux.

Attention: La liste copper est bien connue pour être difficile à programmer, et si vous ne savez pas précisément ce que vous faites, il est très probable que vous bloquiez votre Amiga. Avant de vous embarquer dans des essais sur votre copper, nous vous conseillons de lire un des nombreux ouvrages de référence sur ce sujet. Vous pouvez trouver une bonne explication dans la Bible de l'Amiga (Micro Applications).

COPPER OFF *(Désactiver la liste copper standard)*

COPPER OFF

Cette commande bloque la liste copper courante d'AMOS et désactive complètement l'image. Vous pouvez maintenant créer votre propre image en utilisant une série d'instructions COP MOVE et COP WAIT.

Toutes les listes copper définies par l'utilisateur sont limitées par défaut à un maximum de 12k. Chaque instruction consomme en moyenne deux octets. Vous pouvez donc loger environ six mille instructions. Vous pouvez accroître ce nombre au besoin en utilisant une option spéciale de l'utilitaire CONFIG.

Notez que toutes les instructions copper sont écrites sur une liste logique indépendante qui n'est pas affichée à l'écran. Ceci empêche votre programme d'altérer l'image pendant que la liste copper est en train d'être créée. Pour activer votre nouvel écran, il vous faudra permuter la liste *physique* et la liste *logique* à l'aide de la commande COP SWAP.

Il est également important que vous produisiez vos listes copper dans un ordre bien déterminé, en commençant en haut et à gauche de l'écran et en progressant vers le bas jusqu'en bas à droite. Référez-vous au fichier **EXEMPLE 10.15** du dossier MANUEL pour une démonstration.

COPPER ON *(Redémarrer la liste copper)*

COPPER ON

La commande COPPER ON redémarre les calculs de la liste copper et affiche les écrans AMOS courants. Si vous n'avez encore rien dessiné après l'exécution de l'instruction COPPER OFF, l'écran sera réinitialisé.

COP MOVE *(Ecrire une instruction MOVE dans la liste copper logique)*

COP MOVE adr,valeur

Produit une instruction MOVE dans la liste logique copper.

- *adr* est une adresse d'un registre de 16 bit devant être changé. Elle doit se trouver dans la ZONE DE DONNEES ($7F-$1bE) copper normale. valeur est un nombre entier ayant la taille d'un mot et il doit être chargé dans le registre demandé.

COP MOVEL *(Ecrire une instruction MOVE longue dans la liste copper)*

COP MOVEL adr,valeur

Elle est identique à la commande standard COP MOVE, sauf que *adr* se rapporte maintenant à un registre copper de 32 bits. *valeur* contient un nombre entier représentant un long mot.

COP WAIT *(Instruction Copper WAIT)*

COPY WAIT x,y[,masque x, masque y]

La commande COP WAIT écrit une instruction WAIT dans la liste copper. Le copper attend que les coordonnées machine x,y soient trouvées et redonne la main au processeur principal.

Notez que la ligne 255 est automatiquement gérée par AMOS. Vous n'avez donc pas à vous inquiéter du tout à son propos.

masque x et *masque y* sont des mode points vous permettant d'attendre qu'une certaine combinaison de bits exprimés dans les coordonnées de l'écran soit définie. Les deux masques sont affectés à $1FF par défaut.

COP RESET *(Redéfinir la flèche de la liste copper)*

COP RESET

La commande COP RESET restaure l'adresse utilisée par l'instruction copper suivante au début de la liste copper.

=COP LOGIC *(Adresser la liste copper)*

adr=COP LOGIC

La fonction COP LOGIC renvoie l'adresse absolue dans la mémoire de la liste copper logique. Ceci vous permet d'écrire directement vos instructions COPPER dans la mémoire intermédiaire, en utilisant peut- être un langage assembleur.

Conseils et suggestions

- Avant de créer un écran avec une liste copper définie par l'utilisateur, il faudra d'abord affecter de la mémoire pour les mode points appropriés. Bien que vous puissiez utiliser RESERVE à cette fin, il est bien plus facile de définir un écran fictif au moyen de la

commande SCREEN OPEN. Les registres copper peuvent être chargés des adresses des mode points requis grâce à la fonction LOGBASE.

Vous pourrez alors avoir accès à votre écran en utilisant toutes les fonctions normales d'AMOS. Afin de réserver la quantité correcte de mémoire, définissez le nombre de couleurs en choisissant le nombre maximal utilisé dans le nouvel écran. Ceci peut vous sembler inutile mais simplifie énormément les choses.

- Il est parfaitement possible de combiner les écrans définis par l'utilisateur aux bobs AMOS. Toutefois, si vous utilisez deux mémoires intermédiaires, il vous faudra définir une liste copper séparée pour l'écran logique et l'écran physique. Cette liste peut être obtenue en utilisant la procédure suivante:

  1 Définissez la liste copper du premier écran
  2 Permutez la liste copper logique et la liste copper physique à l'aide de COP SWAP.
  3 Permutez l'écran physique et l'écran logique à l'aide de SCREEN SWAP.
  4 Définissez la liste copper du deuxième écran

Cette procédure garantit la correcte actualisation de vos bobs sur vos nouveaux écrans. Toutes les commandes AMOS normales, AMAL comprise, peuvent être utilisées.

11 Les sprites machines

Un des plus grands attraits du Commodore Amiga est sa capacité à produire des jeux de première qualité qui rivalisent avec ceux rencontrés sur des vraies machines d'arcade. D'excellents programmes comme Battle Squadron et Eliminator en donnent une très bonne illustration.

Pour la première fois, toutes ces fantastiques fonctions sont entre vos mains! AMOS Basic vous permet d'exercer un contrôle total sur les sprites graphiques et les sprites machine. Vous pouvez manoeuvrer ces sprites sans effort au moyen du langage d'animation intégré AMAL; vous n'avez donc pas à être un génie en programmation machine pour créer vos propres fabuleux jeux d'arcade.

Les sprites machine sont des images distinctes pouvant automatiquement se chevaucher à l'écran de votre Amiga. La flèche de la souris est un exemple classique d'un sprite machine. Elle est complètement indépendante de l'écran et elle s'utilise parfaitement dans tous les modes graphiques de l'Amiga.

Puisque les sprites ne perturbent pas l'arrière-plan de l'écran, ils sont parfaits pour le déplacement des objets qui est nécessaire dans un jeu d'arcade. Ils sont non seulement rapides comme des flèches mais ils consomment également très peu de mémoire. Ainsi, lorsque vous écrivez un jeu d'arcade, vous devez toujours placer les sprites machine en haut de votre liste.

Vous pouvez inscire chaque sprite dans une grille de 16 par 255 pixels. Le hardware de l'Amiga supporte des huit sprites de quatre couleurs ou quatre types de sprites de quinze couleurs. La couleur numéro zéro équivaut au transparent, ce qui explique les étranges palettes de couleur.

Au premier abord, ces fonctions ne semblent pas particulièrement impressionnantes. Mais il existe deux ou trois trucs qui peuvent faire augmenter le numéro et la taille de ces sprites au point de les rendre impossibles à identifier.

Un d'entre eux est de prendre chaque sprite machine et de le diviser en un certain nombre de segments horizontaux. Vous pouvez positionner ces segments à des endroits distincts, ce qui vous permet d'afficher clairement et simultanément des douzaines de sprites à l'écran. Similairement, la largeur peut être étendue en construisant un objet à partir de plusieurs sprites particuliers. En utilisant cette technique, il est facile de produire des objets de 128 pixels de largeur.

Jusqu'à ces derniers temps, la seule manière d'exploiter ces techniques était d'utiliser le monde mystérieux du langage assembleur 68000. Vous serez donc content de découvrir qu'AMOS Basic gère automatiquement l'intégralité du procédé! Une fois que vous avez conçu vos sprites avec l'éditeur de sprites AMOS, vous pouvez les manipuler sans effort à l'aide d'une seule instruction Basic.

Les commandes sprites

Vous devez d'avoir une banque sprite chargée en mémoire lorsque vous essayez les diverses commandes de ce chapitre. Nous vous conseillons d'utiliser le fichier SPRITES.ABK de la disquette de données AMOS.

SPRITE *(Afficher un sprite machine à l'écran)*

SPRITE n,x,y,i

La commande SPRITE affiche un sprite machine à l'écran aux coordonnées *x,y* en utilisant le numéro d'image *i*.

N est le numéro d'identification du sprite pouvant être compris entre 0 et 63. Vous pouvez faire correspondre chaque sprite à une image particulière de la banque sprite afin de pouvoir utiliser la même image pour plusieurs sprites.

x et *y* représentent la position du sprite en termes de coordonnées machine. Toutes les mesures sont prises à partir du point chaud de vos images. Ce dernier fait en quelques sortes office de "prise en main" *du sprite* et il est utilisé comme point de référence pour les coordonnées. Normalement, le point chaud est fixé au coin supérieur gauche d'une image. Toutefois, vous pouvez le changer dans votre programme en utilisant la commande HOT SPOT.

Les coordonnées machine sont indépendantes du mode écran et commencent en fait à partir de (-129,-45) sur l'écran implicite. AMOS vous offre plusieurs fonctions intégrées pour effectuer des conversions entre les coordonnées machine et les coordonnées écran qui sont plus faciles à utiliser. Voir les commandes X HARD, Y HARD, X SCREEN et Y SCREEN pour plus de précisions.

I est le numéro d'une certaine image conservée dans la banque sprite. Vous pouvez créer cette banque en utilisant l'éditeur de sprites AMOS et le sauvegarder automatiquement avec votre programme Basic. Vous pouvez également le charger directement par l'instruction LOAD. En outre, vous pouvez utiliser la commande GET SPRITE pour saisir immédiatement une image depuis l'écran courant.

Vous pouvez choisir d'omettre *x,y* et *i* mais vous **devez** mettre les virgules à leur place. Par exemple:

```
Load "AMOS_DATA: Sprites/Octopus.abk"
Sprite 8,200,100,1
Sprite 8,,150,1
Sprite 8,300,,
```

Pour une démonstration des sprites en action, chargez le fichier **EXEMPLE 11.1** du dossier du MANUEL de la disquette de données AMOS.

Les sprites calculés

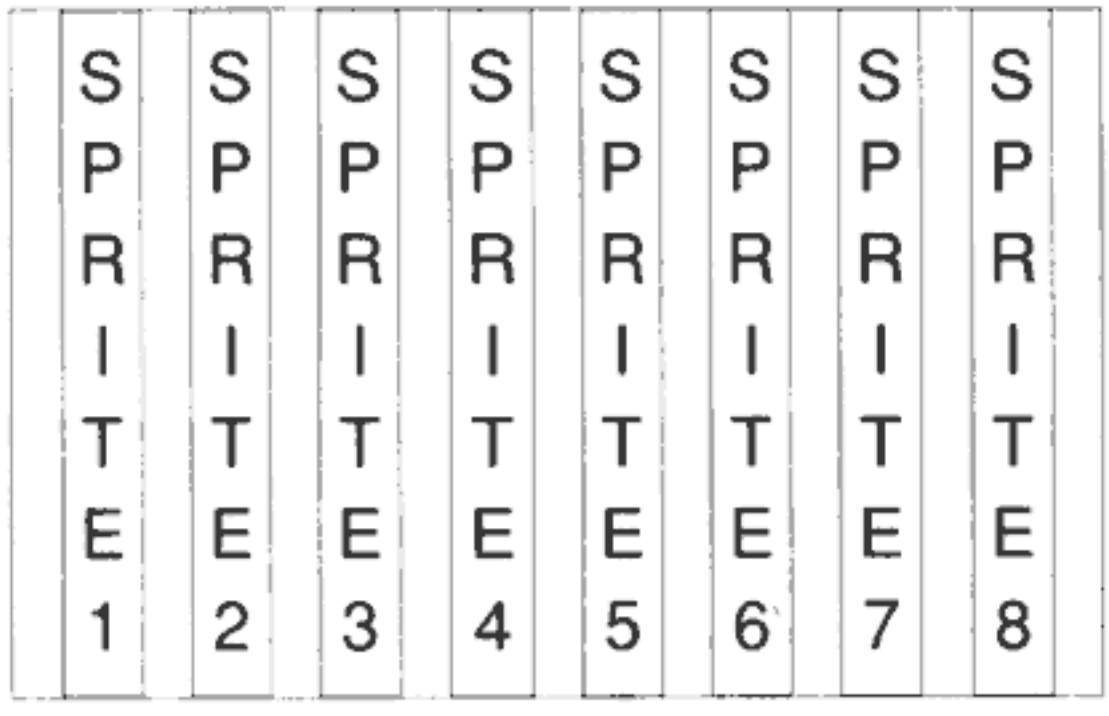
Bien que l'Amiga vous mette en main huit sprites réels, vous pouvez en fait les utiliser pour afficher simultanément 64 objets différents à l'écran. Ces objets sont connus sous le nom de sprites calculés et sont entièrement gérés par AMOS Basic. Vous pouvez affecter les *sprites calculés* d'un nombre supérieur à sept en appelant la commande SPRITE. Par exemple:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Sprite 8,200,100,1
```

La taille d'un sprite calculé provient directement des données de l'image, sa largeur et sa hauteur pouvant varier respectivement entre 16 et 128 pixels et entre 1 et 255 pixels.

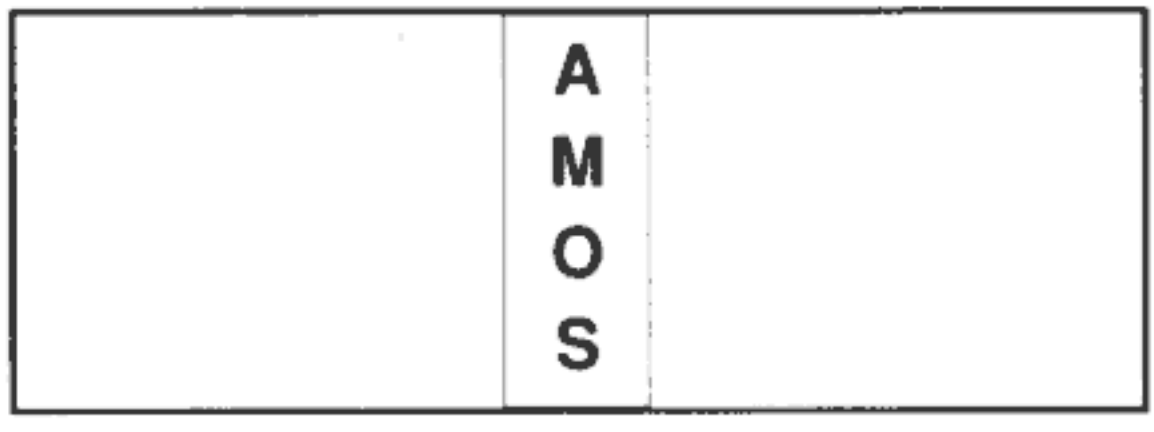
Avant que vous puissiez utiliser pleinement ces sprites, il vous faut comprendre quelques-uns des principes se cachant derrière eux. Chaque sprite machine est représenté par une mince bande d'une largeur de 16 pixels et d'une profondeur d'au plus 256 pixels. Vous pouvez afficher simultanément quatre ou huit bandes à l'écran en fonction du nombre de couleurs.

LES SPRITES MACHINE



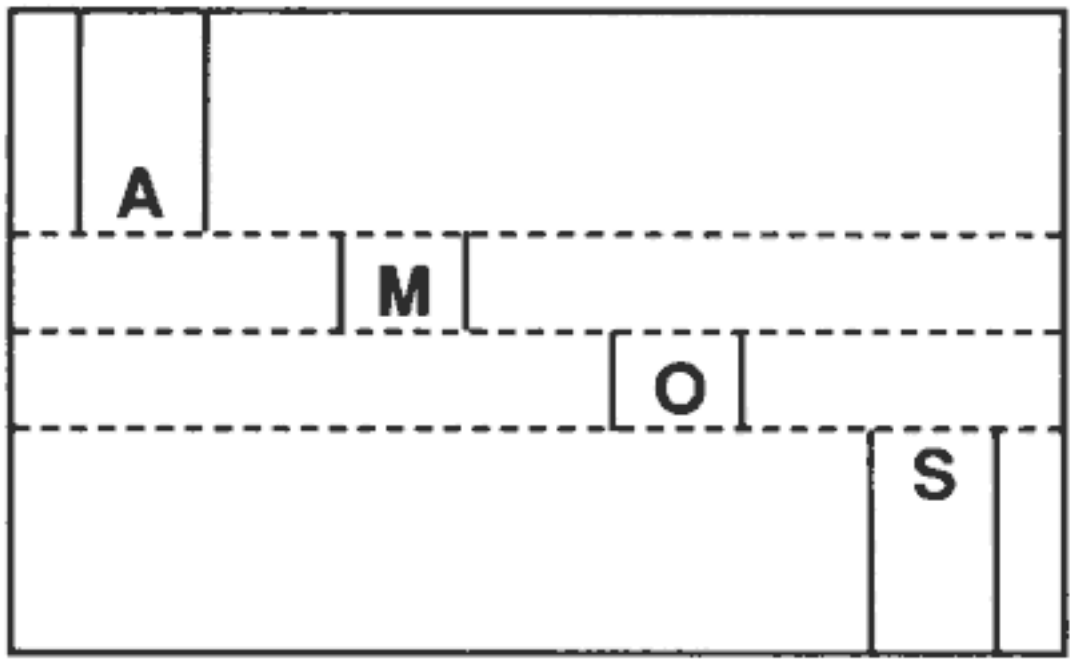
Vous vous êtes sans doute rendu compte que la plus grande partie de la zone se trouvant autour de ces sprites est en fait gaspillée. Cela s'explique par le fait que peu de programmes ont besoin de sprites d'une hauteur supérieure à 40 ou 64 pixels. Cependant, il existe un simple truc nous permettant de récupérer cet espace pour produire des douzaines d'objets supplémentaires à l'écran. Examinez le sprite ci-dessous contenant les lettres A, M, O et S.

Un seul sprite machine



Vous pouvez diviser le sprite en quatre segments horizontaux, chacun d'eux contenant une seule lettre. Le hardware de l'Amiga vous permet de placer chaque section à n'importe quelle position de la ligne courante, ce qui fait un total de quatre sprites calculés. Voici un schéma illustrant ce procédé.

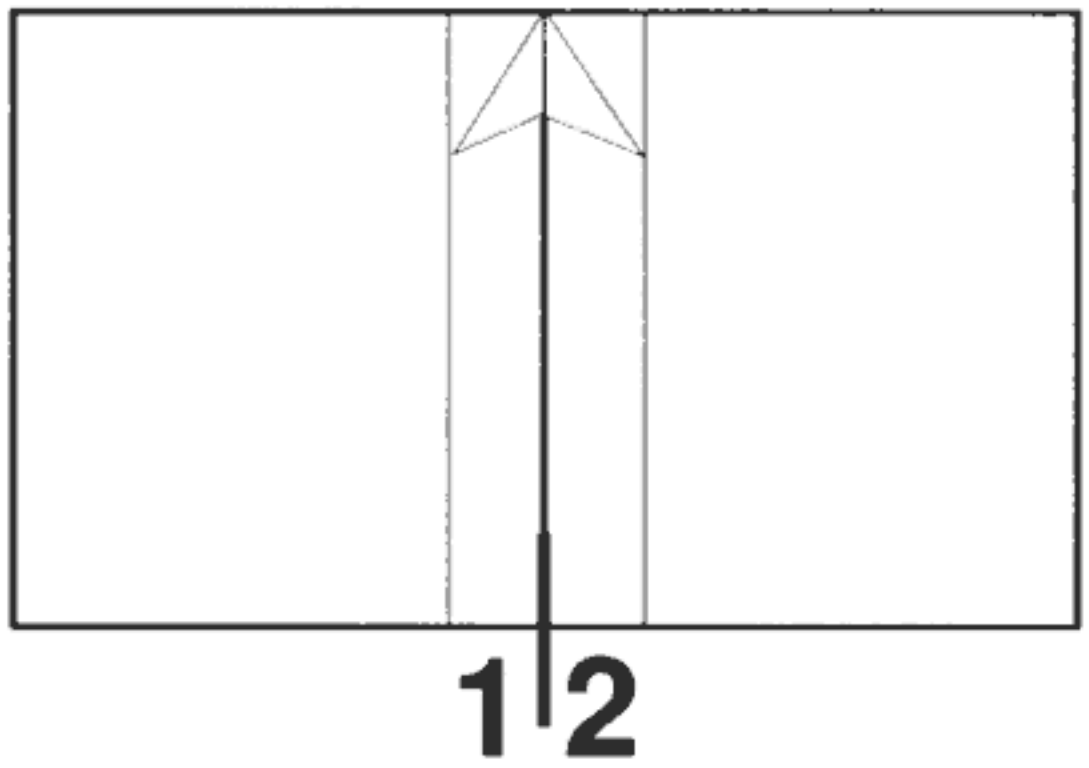
Diviser un sprite machine en des sprites calculés



Comme vous pouvez le constater, les sprites calculés représentent vraiment une seule partie du sprite machine et ils sont affichés horizontalement à différentes positions. Remarquez la ligne séparant chaque objet. C'est un effet secondaire inévitable du procédé de repositionnement produit par le hardware de l'Amiga.

En raison de la façon dont les sprites calculés sont produits, deux ou trois restrictions limitent leur utilisation. Premièrement, vous ne pouvez pas avoir plus de huit sprites calculés par ligne. En pratique, le système est compliqué par la nécessité de produire des sprites dont les dimensions sont supérieures au maximum de 16 pixels. AMOS produit ces objets en positionnant automatiquement plusieurs sprites calculés côte à côte. Le schéma ci-dessous vous en donne une représentation:

Combiner deux sprites



Sprites Machine

Le nombre maximal de huit sprites machine impose donc une limite stricte au nombre des objets que vous pouvez afficher sur une ligne horizontale. La largeur totale des objets ne doivent pas dépasser:

16*8=128 pixels pour les sprites à trois couleurs

16*4=64 pixels pour les sprites à 15 couleurs

Si vous essayez de ne pas vous conformer à cette limitation, vous n'obtiendrez pas un message d'erreur mais votre sprite calculé ne sera pas affiché à l'écran. Par conséquent, vous devez absolument vous assurer de ne jamais enfreindre cette limitation. Pour cela, utilisez la procédure suivante:

Ajoutez les largeurs de tous vos sprites calculés, en multipliant les dimensions de tous les sprites à quinze couleurs par deux. Si vous obtenez un total supérieur à 128, il vous faudra espacer vos sprites afin que leur largeur combinée soit inférieure à cette valeur. Faites bien attention en animant vos sprites avec AMAL car vous découvrirez certaines combinaisons seulement après avoir expérimenté cette séquence pendant quelques temps. Ces problèmes seront dévoilés par la disparition aléatoire d'un ou plusieurs sprites à l'écran.

Même en envisageant le pire, il vous faudra remplacer quelques-uns de vos plus grands sprites à l'aide de la commande Blitter Objects. Ceci augmentera sensiblement la taille globale de votre programme mais influencera très peu la qualité finale de votre jeu.

Ces restrictions ne sont évidemment pas limitées à AMOS Basic. Ils s'appliquent aussi bien sur tous les jeux de l'Amiga, même s'ils ont été écrits en code machine! Il n'y a donc rien vous empêchant de produire votre propre clone Xenon II en utilisant exactement des techniques identiques.

Notez que le sprite machine numéro zéro est normalement affecté à la flèche de la souris. Vous pouvez désaffecter ce sprite par un simple appel à la commande HIDE. Voir le fichier **EXEMPLE 11.2.**

Créer un sprite machine particulier

Le seul vrai problème se posant avec les sprites calculés est que vous ne savez jamais précisément quel sprite machine va être utilisé dans un objet particulier. En général, les sprites machine utilisés dans un objet changeront si ce dernier est déplacé. Ceci peut être quelques fois gênant, surtout lorsque vous animez des objets tels que des missiles qui doivent rester visibles dans de nombreuses situations.

Dans ce cas, il est utile de pouvoir procéder à une affectation directe d'un sprite machine. Vous pouvez permettre l'affectation de sprites machines particuliers en utilisant l'instruction SPRITE et un numéro d'identification compris entre 0 et 7. Exemple:

```
Sprite 1,100,100,2
```

Ceci attribue l'une image numéro 2 au sprite machine numéro 1 . *N* correspond maintenant au numéro d'un seul sprite machine pouvant être compris entre 0 et 7. Si la largeur de votre image est supérieure à seize pixels, AMOS saisira automatiquement le nombre de sprites nécessaire dans un ordre consécutif, en commençant à partir du

sprite que vous avez choisi. Par exemple:

Sprite 2,200,100,1

Supposons que l'image numéro 1 représente une image tricolore de 32 bits. Cette commande affecterait les sprites machine 2 et 3 à cette image. Rien ne se passerait si vous deviez maintenant essayer d'afficher le sprite machine 3 à l'aide d'une commande comme SPRITE 3,150,100,1 parce que ce sprite a déjà été utilisé. Vous n'auriez accès qu'aux sprites 0,1,4,5,6 et 7 et les plus grands numéros et les plus grandes tailles de vos sprites calculés seraient réduits en conséquence.

Chaque sprite de 15 couleurs est mis en place en utilisant un couple de deux sprites tricolores. Toutefois, il n'est pas possible de combiner deux sprites quelconques de cette façon. Seules les combinaisons 0/1,2/3,4/5,6/7 sont permises. Par conséquent, vous devez toujours affecter vos sprites machine en utilisant des numéros de sprite pairs, sinon AMOS commencera votre sprite à partir du groupe de deux sprites suivant, en gaspillant en fait le premier sprite.

Notez également que si vous essayez de créer un grand sprite à quinze couleurs avec ce système, vous pouvez facilement utiliser tous les sprites disponibles dans un seul objet.

Attention! Si vous écrivez un jeu avec des scrollings, il se peut que vous rencontriez des problèmes d'utilisation de sprites en combinaison avec les commandes SCREEN OFFSET et SCREEN DISPLAY. Ces dernières produisent un conflit d'accès direct à la mémoire entre le système des sprites et les modes point de l'écran; elles peuvent également amener de temps en temps des effets indésirés à l'écran.

Ce problème se produit seulement si vous utilisez les sprites machine 6/7. Lorsque vous déplacez l'écran vers la gauche au moyen de SCREEN OFFSET, la durée d'actualisation de votre sprite est réduite puisque l'écran d'accès direct à la mémoire a priorité sur le système sprite. Malheureusement, le temps d'exécution est insuffisant pour dessiner les sprites 6/7 et leur visualisation sera donc altérée.

Pour éliminer ce problème, créez les sprites 6/7 sous la forme de sprites machine particuliers et placez-les en dehors de l'écran en prenant des coordonnées négatives. Ceci empêchera AMOS Basic de les utiliser dans vos sprites calculés. Tout marchera bien à condition que les sprites 6/7 ne soient jamais affichés à l'écran lors de vos scrollings.

La palette de sprites

Les couleurs requises par un sprite machine sont conservées dans les registres de couleurs 16 à 31. A condition que le mode de votre écran courant n'utilise pas ces registres, les couleurs de vos sprites seront distinctes des couleurs de l'écran. Ce qui est très intéressant, c'est que cette condition est également valable dans le mode Ham à 4096 couleurs. Il n'y a donc rien vous empêchant de produire des jeux délirants avec ce système!

Cependant, vous rencontrerez de véritables problèmes lorsque vous utiliserez les écrans à 32 ou 64 couleurs avec les sprites tricolores. Cela s'explique par le fait que les couleurs utilisées par ces sprites sont regroupées de la façon suivante:

Sprites machine	Registres couleur
0/1	17/18/19
2/3	21/22/23
4/5	25/26/27
6/7	29/30/31

Les registres de couleur 16,20,24 et 28 sont considérés comme transparents.

La difficulté provient de la façon dont AMOS produit des sprites calculés. Les sprites machine utilisés pour réaliser ces objets varient au cours du jeu. Ainsi, il est important de donner une même valeur aux trois couleurs utilisées par chaque sprite particulier sinon les couleurs de vos sprites calculés changeront inopinément. Voici une petite procédure AMOS qui effectuera automatiquement tout le procédé pour vous.

```
Procedure INIT_SPRITES
    Get Sprite Palette
    For S=0 To 3
        For C=0 To 2
            Colour S*4+C+17,Colour(C)
        Next C
    Next S
Endproc
```

La restriction ci-dessus ne s'applique évidemment pas aux sprites à quinze couleurs. Si vous voulez utiliser au mieux le mode Extra Half Bright ou le mode à 32 couleurs, vous trouverez peut-être qu'il est plus facile d'éviter de se servir des sprites à quatre couleurs.

GET SPRITE PALETTE *(Saisir les couleurs des sprites pour mettre à l'écran)*

GET SPRITE PALETTE [masque]

Cette commande charge l'intégralité de la palette de couleurs utilisée pour vos images de sprites dans l'écran courant. Le masque optionnel vous permet de charger simplement un choix de couleurs à partir de la palette de sprites. Chacune des 32 couleurs est représentée par un seul bit dans le masque, numéroté de droite à gauche. Le bit d'extrême droite représente l'état de la couleur zéro, le suivant la couleur 1, etc... Pour charger une couleur, il suffit simplement de forcer le bit approprié à 1. Par exemple, si vous vouliez juste copier les quatre premières couleurs, vous définiriez la configuration binaire de la manière suivante:

```
Sprite Palette %0000000000001111
```

A propos, puisque les bobs utilisent la même banque sprite que les sprites eux-mêmes, vous pouvez également utiliser cette commande pour charger les couleurs d'un bob.

Contrôler les sprites

SET SPRITE BUFFER *(Définir la hauteur des sprites machine)*

SET SPRITE BUFFER n

Cette commande définit la zone de travail dans laquelle AMOS crée les images des sprites machine. Les valeurs possibles de *n* sont comprises entre 16 et 256. Pour définir la valeur correcte de *n*, il vous suffit d'examiner les sprites dans l'éditeur de sprites et de trouver le plus grand sprite en termes de longueur. Tout sprite plus grand que *n* sera simplement tronqué au point de troncature approprié.

SET SPRITE BUFFER vous permet de reprendre toute mémoire redondante que votre jeu ou votre application n'utilise pas.

Vous pouvez calculer la quantité de mémoire consommée par le tampon sprite en utilisant la formule suivante:

Mémoire = N*4*8*3 = N*96

Ainsi la taille la plus petite du tampon est de 1536 octets et la plus grande est de 24k. Remarque: Cette commande efface toutes les affectations courantes de sprites et replace le curseur de la souris à sa position d'origine.

SPRITE OFF *(Retirer un ou plusieurs sprites de l'écran)*

SPRITE OFF [n]

La commande SPRITE OFF retire un ou plusieurs sprites de l'écran. Tous les déplacements courants des sprites sont suspendus. Afin de les relancer, il vous faut réinitialiser votre schéma de déplacement.

SPRITE OFF Retire tous les sprites de l'écran

SPRITE OFF n Désactive seulement le sprite *n*

Notez que tous les sprites sont automatiquement désactivés lorsque vous appelez l'éditeur AMOS Basic. Ils seront automatiquement renvoyés à leur position d'origine lors de votre prochain passage au mode direct.

SPRITE UPDATE *(Contrôler les déplacements des sprites)*

SPRITE UPDATE [ON/OFF]

La commande SPRITE UPDATE vous permet d'exercer un contrôle parfait sur le déplacement de vos sprites. D'une manière générale, lorsque vous déplacez un sprite, sa position est actualisée automatiquement lors du prochain signal de fin de balayage (voir WAIT VBL). Mais si vous déplacez beaucoup de sprites au moyen de la commande

SPRITE, les actualisations se feront avant que tous les sprites soient déplacés. Ceci peut donner lieu à un saut perceptible de vos schémas de déplacement. Dans ce cas, vous pouvez désactiver le système d'actualisation automatique en utilisant la commande SPRITE UPDATE OFF.

Dès que vous avez réussi à déplacer vos sprites, vous pouvez alors les faire glisser lentement pour les mettre en place en faisant un appel à SPRITE UPDATE. Cette commande repositionnera tous les sprites que vous avez déplacés depuis la dernière actualisation.

Référez-vous à UPDATE EVERY, UPDATE et BOB UPDATE.

= X SPRITE *(Saisir l'abscisse X d'un sprite)*

x = X SPRITE(n)

Cette commande renvoie l'absisse courante X du sprite n, mesuré en utilisant le système machine. Elle vous permet de vérifier rapidement si un sprite est passé de l'autre côté du bord de l'écran Amiga.

= Y SPRITE *(Saisir l'ordonnée Y d'un sprite)*

y = Y SPRITE(n)

Cette commande renvoie la position verticale d'un sprite. Comme de coutume, n se rapporte au numéro de sprite et peut varier de 0 à 63. Rappelez-vous que toutes les positions de sprites sont mesurées en coordonnées machine. Voir le fichier **EXEMPLE 11.3.**

GET SPRITE *(Charger une portion de l'écran dans la banque sprite)*

GET SPRITE [s] i,x1,y1 To x2,y2

Cette instruction vous permet de saisir directement des images de l'écran et de transformer ces dernières en sprites. Les coordonnées *x1,y1 et x2,y2* définissent une zone rectangulaire qui doit être saisie dans la banque sprite. En général, toutes les images sont prises depuis l'écran courant. Toutefois, il est également possible de s'emparer de l'image d'un écran spécifique en utilisant l'écran optionnel numéro *s*.

Remarque: Aucune limitation n'est apportée à la zone devant être saisie de cette façon. Tout marchera bien à condition que vos coordonnées se trouvent à l'intérieur des frontières de l'écran.

I représente le numéro de la nouvelle image. S'il n'existe pas de sprite doté de ce numéro, une nouvelle image sera automatiquement créée. AMOS prendra également la peine de lui réserver une banque sprite si celle-ci n'a pas été préalablement défini. Référez-vous au fichier **EXEMPLE 11.4.**

Il existe également une instruction équivalente à GET BOB qui est exactement identique à GET SPRITE. Puisque la banque sprite est partagée entre les bobs et les sprites, le format des images est exactement le même. Il est donc parfaitement possible

d'utiliser les deux instructions avec les bobs ou les sprites. Essayez de modifier l'instruction sprite de l'exemple précédent pour obtenir:

```
Bob 1,0,0,1
```

Les fonctions de conversion

=X SCREEN =Y SCREEN *(Convertir des coordonnées machine en coordonnées écran*

X=X SCREEN([n,] xcoord)
Y=Y SCREEN([N,] ycoord)

Les fonctions convertissent une coordonnée machine en une coordonnée écran se rapportant à l'écran courant. Si les coordonnées machine se trouvent en dehors de l'écran, alors les deux fonctions renverront des déplacements relatifs depuis la frontière des écrans. Tapez ce qui suit en mode direct:

```
Print X Screen(130)
```

Le résultat sera -2. Cela s'explique par le fait que l'abscisse écran 0 est égale à l'abscisse machine 128 et par conséquent, la conversion de l'abscisse 130 en une abscisse écran donne une position de deux pixels sur la gauche de l'écran.

Si le numéro optionnel d'écran est précisé, alors les coordonnées se rapportant à l'écran *n* seront renvoyées.

X HARD *(Convertir des coordonnées écran en coordonnées Y HARD machine)*

X = XHARD ([n,] absc.x)

Ces fonctions convertissent une abscisse écran en une abscisse machine. Il existe quatre fonctions de conversion, la syntaxe ci- dessus convertissant *absc.x*, une abscisse se rapportant à l'écran courant en une abscisse machine.

Y = YHARD ([n,] ord.y)

Convertit une ordonnée Y se rapportant à l'écran courant en une ordonnée machine. Comme auparavant, n spécifie le numéro d'écran devant être utilisé avec ces fonctions. Toutes les ordonnées se rapportant à cet écran seront renvoyées.

=I SPRITE *(Renvoyer l'image courante d'un sprite)*

Image=I Sprite(n)

Cette fonction renvoie le numéro d'image courante en cours d'utilisation par le sprite n. Une valeur zéro sera signalée si le sprite ne s'affiche pas.

12 Les objets graphiques à manipuler

Les sprites machine sont certainement puissants mais leur usage est limité par deux ou trois ennuyeuses restrictions. Non seulement le nombre possible de sprites par ligne horizontale est de huit mais chacun des sprites est doté d'au plus 15 couleurs.

Pour contourner ces limitations, nous allons recourir au blitter de l'Amiga. Celle-ci est capable de copier des images à l'écran à une vitesse de près d'un million de pixels par seconde! Avec elle, on peut créer ce qu'on appelle des bobs (ou objets graphiques à manipuler).

Vous pouvez déplacer les bobs, comme les sprites, indépendamment de l'écran sans détruire les graphiques existants. Mais à la différence des sprites, les bobs sont mémorisés comme une partie de l'écran courant; vous pouvez donc les créer dans n'importe quel mode graphique avec au plus 64 couleurs. En outre, le nombre de bobs que vous pouvez afficher à l'écran est seulement limité par la mémoire disponible.

Bien évidemment, toute cette puissance ne s'obtient pas gratuitement. En pratique, les bobs sont légèrement plus lents que les sprites et ils consomment bien davantage de mémoire. Vous devez donc faire un compromis en choisissant la vitesse des sprites ou la flexibilité des bobs. Heureusement, il n'y a rien qui vous empêche d'utiliser à la fois les bobs et les sprites dans le même programme. C'est l'approche qui a été adoptée dans la plupart des jeux du commerce. Les bobs sont utilisés pour les grands objets comme les vaisseaux spaciaux alors que les sprites machine sont souvent réservés pour les petits objets se déplaçant rapidement tels que les missiles.

BOB *(Tracer un objet graphique sur l'écran courant)*

BOB n,x,y,i

La commande BOB crée un bob *n* aux coordonnées *x,y* en utilisant l'image numéro *i*.

n est le numéro d'identification du bob. Les valeurs autorisées sont normalement comprises entre 0 et 63, mais le numéro des bobs peut être élevé en utilisant une option se trouvant dans le programme de configuration AMOS. A condition que vous ayez une mémoire suffisante, vous pouvez fixer cette limite à n'importe quel numéro de bob que vous souhaitez.

x et *y* spécifient la position du bob en utilisant les coordonnées écran standard. Ces coordonnées ne sont pas les mêmes que les coordonnées machine utilisées par la commande équivalente SPRITE. Comme les sprites, chaque bob est contrôlé au moyen d'un point chaud que vous pouvez changer à tout moment à l'aide de la commande *HOT SPOT*.

i se rapporte à une image qui doit être affectée au bob de la banque sprite. Le format de cette image est identique à celle utilisée par les sprites; vous pouvez donc

vous servir des mêmes images pour les sprites ou les bobs.

Après avoir crée un bob, vous pouvez changer indépendamment sa position ou son apparence en omettant un ou plusieurs paramètres de cette instruction. N'importe quel des numéros *x,y* ou *image* peut être mis de côté avec les paramètres manquants renfermant leur valeur d'origine. Ceci est particulièrement utile si vous animez votre bob avec le langage AMAL, puisque ce langage vous permet de déplacer votre objet où vous voulez sans perturber votre présente séquence d'animation. Toutefois, vous devez toujours placer les virgules à leur position d'origine, sinon un message d'erreur de syntaxe vous sera communiqué. Par exemple:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Flash Off : Get Sprite Palette
Channel 1 To Bob 1
Bob 1,0,100,1
Amal 1,"Anim 0,(1,4)(2,4)(3,4)(4,4)"
Amal On
For X=1 To 320
   Bob 1,X,,
   Wait Vbl
Next X
```

Si vous déplacez un bob, l'arrière-plan est replacé à sa position d'origine, en produisant un effet identique à celui de la commande équivalente SPRITE. A la différence de STOS sur le ST, chaque objet est doté de sa propre zone de rangement. Ceci réduit la quantité de mémoire utilisée par les bobs et améliore considérablement la vitesse globale d'exécution. A cause de la manipulation d'objets, on ne peut pas établir de véritables comparaisons entre les sprites STOS et les bobs AMOS.

Bien que la commande BOB convienne à un petit nombre de bobs, un scintillement agaçant apparaît lorsque vous essayez d'utiliser plus de trois ou quatre objets à l'écran. Vous pouvez donc voir les bobs se tracer ce qui donne un effet brillant désagréable.

Une des solutions permettant d'améliorer la qualité de vos animations est de dessiner simplement les bobs dans le quart inférieur de l'écran. Puisque les bobs sont retracés très rapidement, les actualisations peuvent souvent être terminées avant que la partie inférieure de l'écran soit visualisée. Ceci vous donne des déplacements assez graduels tout en consommant très peu de mémoire; par conséquent, c'est une astuce utile si vous êtes à court de place. Référez-vous au fichier **EXEMPLE 12.1.**

Evidemment, cette solution répondant à ce problème de scintillement ne peut pas être considérée comme brillante. Par conséquent, avant que vous jetiez votre disquette AMOS avec dégoût, vous serez soulagé d'apprendre qu'il existe une façon simple d'éliminer complètement ce clignotement, même si vous utilisez des douzaines de bobs à n'importe quel endroit de l'écran.

DOUBLE BUFFER *(Créer un double tampon écran)*

DOUBLE BUFFER

La commande DOUBLE BUFFER crée une deuxième copie invisible de l'écran courant. Toutes les opérations graphiques, y compris les déplacements de bobs, sont maintenant réalisées directement dans *l'écran logique*, sans perturber du tout votre image télé. Une fois que l'image a été retracée, l'écran logique s'affiche et l'écran *physique* d'origine devient le nouvel écran logique. Le procédé tout entier se répète d'une façon permanente, en produisant une image inébranlable même lorsque vous déplacez des centaines de bobs à la fois.

L'intégralité de la procédure est automatiquement effectuée par AMOS Basic; par conséquent, après que vous ayez effectué cette instruction, vous pouvez tout à fait l'oublier. Notez que puisque les sprites machine sont toujours affichés au dessus à l'écran physique courant, ce système n'aura absolument aucun effet sur toutes les animations sprite présentes.

Le double buffer marche tout aussi bien dans tous les modes graphiques d'Amiga. Vous pouvez l'utiliser avec le dual playfield. Mais faites attention: Le double buffer double la quantité de mémoire utilisée dans vos écrans. Si vous essayez de passer en double tampon un trop grand nombre d'écrans, vous allez rapidement être à court de mémoire. Référez-vous au fichier **EXEMPLE 12.2.**

En fait, le double tamponnage est une technique incroyablement utile et vous pouvez vous en servir pour la plupart des types de jeu. Vous la retrouverez dans pratiquement tous les jeux du commerce, comme dans Starglider; c'est pourquoi elle est une partie intégrante de l'AMOS Basic. Vous trouverez une explication détaillée de ce procédé dans le chapitre ECRANS. Voir également les commandes SCREEN SWAP et AUTOBACK.

SET BOB *(Définir le mode de tracé des bobs)*

SET BOB n,back,planes,minterms

La commande SET BOB change le mode de tracé utilisé pour afficher un objet de manipulation d'objet à l'écran. *n* est le numéro du bob que vous souhaitez affecter.

back choisit le type de tracé de l'arrière-plan se trouvant derrière votre bob. Trois cas se présentent:

- Une valeur de zéro indique que la zone se trouvant sous votre bob devrait être sauvegardée en mémoire. Les données de l'image précédente sont automatiquement remplacées si le bob est déplacé, ce qui permet un déplacement graduel.
- Si le paramètre *back* est positif, alors l'arrière-plan d'origine est supprimé et la zone se trouvant derrière le bob sera remplie de la couleur *back - 1*. Ce système est parfait pour déplacer les bobs sur une zone en couleur pleine telle qu'un ciel tout bleu, puisqu'il est bien plus rapide que le système standard de tracé.
- Désactivez le procédé de retracé en attribuant à back une valeur négative telle que -1. Vous pouvez alors désactiver le procédé automatique d'actualisation en utilisant BOB UPDATE et déplacer manuellement les bobs en faisant appel à BOB DRAW. Cette commande vous permet de regénérer l'arrière-plan de l'écran en utilisant vos propres routines personnalisées de tracé.

planes est une configuration binaire indiquant à AMOS les plans de l'écran sur lesquels votre bob sera tracé. Vous savez probablement que l'écran de l'Amiga est divisé en un

certain nombre de plans binaires distincts. Chaque plan définit un seul bit dans la couleur finale visualisée à l'écran.

Le premier plan est représenté par le bit un, le second par le bit deux, etc... De manière générale, le bob est tracé dans tous les plans binaires du mode de l'écran courant. Ceci correspond à une configuration binaire de %111111.

En attribuant une valeur de zéro à certains de ces bits, vous pouvez omettre les couleurs sélectionnées de vos bobs lors de leur tracé. Vous pouvez ainsi obtenir des effets fascinants à l'écran.

minterm sélectionne le mode de manipulation utilisé pour tracer vos bobs à l'écran. Vous trouverez une complète description des modes possibles de tracé dans le chapitre COPIE D'ECRAN.

minterm est généralement initialisé à l'une des deux valeurs suivantes:

%11100010	Si le bob est utilisé avec un masque
%11001010	Si AUCUN MASQUE a été défini

N'hésitez surtout pas à mettre en pratique ces différentes combinaisons. Vous n'avez pas à craindre une panne de votre ordinateur si vous commettez une erreur. Les utilisateurs avertis de l'Amiga peuvent trouver les informations suivantes utiles.

Source de manipulation	**Destination**
A	Masque
B	Objet
C	Ecran de destination

Nous vous recommandons d'utiliser SET BOB **avant** d'afficher vos bobs à l'écran. Si vous ne le faites pas, votre Amiga ne se plantera pas et vous n'obtiendrez pas un message d'erreur, mais votre image risque d'être altérée.

NO MASK *(Retirer le masque de dessin)*

NO MASK [n]

Un masque de dessin est crée par défaut pour chaque bob affiché à l'écran. Le masque est combiné à l'arrière-plan de l'écran pour rendre la couleur zéro transparente. Il est également utilisé par les différentes commandes de détection de collision.

La commande NO MASK retire ce masque et l'image entière se trace automatiquement à l'écran. Toute partie de l'image de couleur zéro s'affichera alors directement sur l'arrière-plan existant.

n est le numéro d'image du masque qui doit être retiré. Ce masque ne devrait jamais être effacé si l'image est active à l'écran, sinon le bob qui lui est associé sera altéré. Si vous devez retirer le masque de cette façon, il est important de désactiver d'emblée les bobs correspondants au moyen de BOB OFF. Voici un exemple:

Centre "Cliquez sur le bouton de la souris pour retirer le masque"

```
Double Buffer : Load "AMOS_Data:Sprites/Octopus.abk" : Get Sprite palette
Do
   Bob 1,X Screen(X Mouse),Y Screen(Y Mouse),1
   If Mouse Click Then Bob Off : No Mask 1
Wait VBL
Loop
```

Voir MAKE MASK

AUTOBACK *(Définir le mode automatique de copie d'écran)*

AUTOBACK n

Lorsque vous utilisez un écran à double tampon, il est important de rendre vos opérations de tracé et les déplacements de vos objets de manipulation graphique synchrones. Rappelez-vous que chaque écran à double tampon consiste de deux images distinctes. Il y a un écran pour l'image courante et une autre pour l'image qui se construit. Si l'arrière-plan se trouvant dessous un bob change alors qu'il est retracé, le contenu de ces écrans sera différent et vous obtiendrez un très fort scintillement qui est gênant.

Le système unique AMOS AUTOBACK résout parfaitement ce problème. Il vous permet de produire des graphiques dans n'importe quel des modes graphiques, en fonction des exigences précises de votre programme. Voici une liste de ces options en ordre inverse.

AUTOBACK 2 *(Mode automatique - défaut)*
Dans ce mode, toutes les opérations de tracé sont automatiquement associées aux actualisations des bobs. Par conséquent, tout ce que vous dessinez à l'écran s'affichera directement dessous vos bobs, comme par magie. Les principes se cachant derrière ce système peuvent être démontrés par la programmation suivante:

```
Rem Tracez sur le premier écran
Rem Retirez les Bobs
Bob Clear
Rem Exécutez les opérations graphiques
Plot 150,100 : Rem Cela peut être n'importe quoi
Bob Draw : Rem Retracez les Bobs
Screen Swap : Rem Prochain Ecran
Wait Vbl
Rem Tracez objet sur prochain écran
Bob Clear
Plot 150,100 : Rem Exécutez votre opération pour la deuxième fois
Bob Draw
Screen Swap : Rem Revenez sur le premier écran
Wait Vbl
```

Comme vous pouvez le constater, toutes les actualisations d'écran sont effectuées deux fois. Une opération est réalisée pour l'écran logique et une pour l'écran physique. Pour une démonstration, référez-vous au fichier **EXEMPLE 12.3 .**

En conséquence, le tracé de vos graphiques prend deux fois plus de temps. En outre, le programme peut être arrêté par deux wait VBL, (ou 2/50ème de seconde), chaque fois que vous sortez quelque chose à l'écran. Ceci peut causer des retards ennuyeux dans l'exécution des opérations de première importance comme la détection de collision.

AUTOBACK 1 *(Mode semi-automatique)*
Cette commande effectue chaque opération graphique simultanément dans l'écran physique et dans l'écran logique. Comme ce mode ne tient pas compte de vos objets graphiques à manipuler, vous devez seulement utiliser ce système pour effectuer des tracés en dehors de la zone courante d'animation.

A la différence du mode automatique, vous n'avez pas besoin d'interrompre votre programme avant le prochain balayage. Le mode 1 est donc parfait pour des objets tels que les panneaux de commandes ou les tableaux de marquage des points qui doivent être constamment mis à jour au cours du jeu.

AUTOBACK 0 *(Mode manuel)*
Cette commande arrête l'exécution du système AUTOBACK. Tous les graphiques sont aussitôt sortis sur l'écran logique à la vitesse la plus élévée possible. Utilisez cette option s'il vous faut retracer de grandes portions de l'arrière-plan de votre écran à maintes reprises au cours d'un jeu. Ceci vous permettra d'effectuer sans risque les routines de détection de collision à des intervalles réguliers, sans supprimer la qualité générale des effets d'animation. Voici une boucle typique de programme que vous pouvez examiner.

```
Bob Update Off : Rem Supprimez les mises à jour automatiques de l'écran
Repeat
Screen Swap
Wait Vbl
Rem Supprimez vos Bobs de l'écran
Bob Clear
Rem
Rem Retracez maintenant n'importe quel de vos graphiques ayant changé
Rem Exécutez vos routines de jeu (Détection de collision, etc...)
Rem Actualisez vos images Bob
Rem
Bob Draw
Rem Permutez les écrans physique et logique
Until WIN : Rem Continuez jusqu'à la fin du jeu
```

Notez que cette procédure est valable seulement s'il existe une progression graduelle d'écran en écran. C'est à vous de voir si vous voulez conserver le contenu des écrans physique et logique en marche l'un avec l'autre. Vous trouverez un exemple de cette technique dans **le fichier EXEMPLE 12.4.**

Supposez par exemple que vous vouliez afficher un bob sur une série de de blocs disposés au hasard. Vous pourriez essayer d'utiliser la routine suivante:

```
Load "AMOS_DATA:Sprites/Octopus.abk" : Flash Off : Get Sprite Palette
Double Buffer : Cls 0 : Autoback 0 : Update Off
Bob 1,160,100,1
Do
   Bob Clear
   X=Rnd(320)+1 : Y=Rnd(200)+1 : W=Rnd(80)+1 : H=Rnd(50)+1 : I=Rnd(15)
   Ink I : Bar X,Y TO X+W,Y+H
   Rem Ce serait normalement un appel à votre routine de détection de collision
   Bob Draw
   Screen Swap : Wait Vbl
Loop
```

Mais puisqu'aucune relation n'est établie entre les écrans physique et logique, l'image commencera à scintiller de façon permanente d'écran en écran. Pour surmonter ce problème, il vous faudra simuler le système AUTOBACK d'origine de la manière suivante:

```
Load "AMOS_DATA:Sprites/Octopus.abk" : Flash Off : Get Sprite Palette
Double Buffer : Cls 0 : Autoback 0 : Update Off
Bob 1,160,100,1
Do
   Rem Actualisez le premier écran
   Screen Swap: Wait Vbl
   Bob Clear
   X=Rnd(320)+1 : Y=Rnd(200)+1 : W=Rnd(80)+1 : H=Rnd(50)+1 : I=Rnd(15)
   Ink I : Bar X,Y TO X+W,Y+H
   Bob Draw
   Rem Actualisez le second écran
   Screen Swap : Wait Vbl
   Bob Clear
   Ink I : Bar X,Y To X+W,Y+H
   Bob Draw
Loop
```

Les deux écrans sont maintenant actualisés et contiennent exactement la même information; l'image restera inébranlable même si beaucoup d'activités se déroulent dans l'arrière-plan.

Vous pouvez utiliser Autoback sans risque dans tout votre programme. Ainsi, il est tout à fait possible d'utiliser des méthodes de tracé différentes dans les différentes portions de votre écran. Cette commande est parfaitement compatible avec toutes les opérations graphiques y compris les blocs, les icônes et le fenêtrage.

Les commandes de contrôle des bobs

BOB UPDATE *(Commander les déplacements des bobs)*

BOB UPDATE [ON/OFF]

En général, tous les bobs sont actualisés une fois toutes les 50èmes de seconde au moyen d'une routine d'interruption intégrée. Bien que cette actualisation soit pratique pour la plupart des programmes, certaines applications demandent un contrôle plus précis du procédé de retracé.

BOB UPDATE OFF désactive les actualisations de bob et désactive toutes les opérations automatiques de basculement de l'écran si celles-ci ont été précédemment sélectionnées. A présent, vous pouvez retracer tous les bobs au point qui convient dans votre programme en utilisant la commande BOB UPDATE. Cette commande est idéale lorsque vous animez un grand nombre d'objets puisqu'elle vous permet de mettre vos bobs en place avant de les tracer à l'écran. Ceci donne forcément des déplacements plus réguliers dans votre jeu.

Un conseil: Les actualisations de bob se produisent seulement au prochain retour de balayage. Notez également que BOB UPDATE retrace toujours les bobs à l'écran logique courant; par conséquent, si vous oubliez d'utiliser la commande SCREEN SWAP, apparemment rien ne se produira.

BOB CLEAR *(Effacer tous les bobs de l'écran)*

BOB CLEAR

L'instruction BOB CLEAR efface tous les bobs actifs de l'écran et retrace les zones d'arrière-plan se trouvant derrière eux. Elle est destinée à être utilisée avec BOB DRAW pour remplacer la commande standard BOB UPDATE.

BOB DRAW *(Retracer les bobs)*

Si vous retracez des bobs à l'écran, il vous faut systématiquement suivre les étapes suivantes:

1. Tous les bobs actifs sont effacés de l'écran LOGIQUE et les zones en arrière-plan sont remplacées. Cette étape est effectuée par BOB CLEAR.
2. Une liste de tous les bobs déplacés est établie depuis l'actualisation précédente.
3. Les zones en arrière-plan sous les nouvelles coordonnées de l'écran sont sauvegardées en mémoire.
4. Tous les bobs actifs sont retracés à leur nouvelle position sur l'écran logique.
5. Si la fonction DOUBLE BUFFER est activée, les écrans physiques et logiques sont permutés.

La commande BOB DRAW exécute directement les étapes 2 et 4 de ce processus. Supposez que vous souhaitiez créer un jeu d'arcade avec des scrollings. Dans ce cas, il

est absolument essentiel que vos scrollings soient parfaitement synchrones avec les effets de déplacement. Si les ennemis devaient se déplacer lors du scrolling, les zones de leur arrière-plan ne seraient pas retracées à la bonne place. Ceci altèrerait complètement votre image et résulterait en un fouilli pas possible à l'écran. Pour obtenir une démonstration de ce procédé, chargez **le fichier EXEMPLE 12.5** du dossier MANUEL.

= X BOB (Renvoyer l'abscisse d'un bob)

x1=X BOB(n)

Cette fonction renvoie l'abscisse courante du bob numéro n. Cette coordonnée est mesurée en rapport à l'écran courant. Voir Y SPRITE, X MOUSE et Y MOUSE.

=Y BOB *(Renvoyer l'ordonnée d'un bob)*

y1 = Y BOB(n)

Y BOB complète la commande X BOB en renvoyant l'ordonnée Y du bob numéro n. La valeur sera renvoyée en utilisant les coordonnées habituelles de l'écran. Voir également à X BOB, X SPRITE, Y SPRITE, X MOUSE et Y MOUSE.

= I BOB *(Renvoyer l'image courante d'un bob)*

Image=I Bob(n)

Cette fonction renvoie le numéro d'image courant en cours d'utilisation par le bob n. Une valeur de zéro sera communiquée si le bob n'est pas affiché.

LIMIT BOB *(Renfermer un bob dans une zone rectangulaire de l'écran)*

LIMIT BOB[n,] x1 TO x2,y2

Cette commande retreint la visibilité de vos bobs à une zone rectangulaire de l'écran délimitée par les coordonnées *x1,y1* et *x2,y2* se rapportant à l'écran courant. Les coordonnées x sont arrondies à la frontière la plus proche de 16 pixels. Notez que la largeur de cette zone doit toujours être supérieure à celle de vos bobs, sinon vous obtiendrez *un message d'erreur appel de fonction interdite.*

S'il est précisé, *n* indique le numéro du seul bob devant être affecté par cette instruction, sinon **tous** les bobs seront délimités. Vous pouvez retrouver la visibilité de l'intégralité de l'écran en tapant:

```
Limit Bob
```

GET BOB (Charger une portion de l'écran dans la banque sprite)

GET BOB [s,] i,x1,y1 To x2,y2

Cett instruction est identique à la commande GET SPRITE. Elle saisit une image dans la banque sprite de l'écran courant.

x1,y1 et x2,y2 sont les coordonnées des angles supérieurs et inférieurs de la zone rectangulaire devant être saisie.

i indique le numéro d'image devant recevoir cette zone. s sélectionne un numéro d'écran optionnel duquel l'image doit être saisie. Voir la commande GET SPRITE pour plus de précisions. Par exemple:

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF",3
Get Sprite 1,0,64 To 320,164
Clw
Bob 1,0,0,1
Wait Vbl
```

Vous pouvez trouver un exemple plus détaillé dans **le fichier EXEMPLE 12.6:** il charge une image en mémoire et vous permet de saisir un bob dans la banque sprite depuis n'importe quelle partie de l'écran. Voir HOT SPOT et MASK.

PUT BOB *(Afficher la copie d'un bob à l'écran)*

PUT BOB n

Cette commande est exactement l'opposé de la précédente GET BOB. L'action de PUT BOB est d'afficher la copie d'un bob numéro *n* à la position présente de l'écran. Ceci est possible en empêchant l'arrière-plan derrière le bob de se retracer pendant la prochaine impulsion de suppression images. Afin de rendre les actualisations de bobs et l'image synchrones, vous devriez toujours faire suivre cette commande d'une instruction WAIT VBL.

Notez qu'après avoir exécuté cette instruction, vous pouvez déplacer ou animer le bob d'origine sans obtenir d'effets indésirés.

PASTE BOB *(Tracer une image de la banque sprite à l'écran)*

PASTE BOB x,y,i

La commande PASTE BOB trace une copie d'une image numéro *i* aux coordonnées **écran** *x,y*. A la différence de PUT BOB, cette image est tracée immédiatement à l'écran et toutes les règles habituelles de découpage sont ici appliquées. Voir PASTE ICON.

BOB OFF *(Supprimer un bob de l'écran)*

BOB OFF [n]

Il se peut que vous souhaitiez supprimer certains bobs de l'écran. La commande BOB

OFF efface le bob numéro n de l'écran et met fin à toutes autres animations correspondantes. Si vous omettez de préciser *n*, tous les bobs seront supprimés par cette instruction. Essayez l'instruction suivante:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Get Sprite Palette
Bob 2,110,110,2
Wait Key
Bob Off 2
Direct
```

Bascule automatique des bobs

Dans un grand nombre de jeux, le personnage principal doit bouger de gauche à droite et de haut en bas. Jusqu'à maintenant, vous étiez obligé de garder, dans la banque des bobs, les copies inversées de petites séquences d'animation pour le même personnage. Puisque le personnage principal est habituellement celui qui bouge le plus, vous perdez une énorme quantité de place!

Prenez le jeu RanXérox (en vente maintenant!) que François a écrit il y a longtemps. Il a produit une routine basculante qui lui a permis de garder seulement une seule copie du personnage principal de la banque. Cette routine a été perfectionnée et placée dans AMOS.

Comment fonctionne-t-elle? Imaginez que votre personnage se dirige vers la gauche en marchant puis revient vers la droite. Votre banque conservera seulement l'image du personnage se dirigeant vers la droite. Pour afficher l'image correcte, il vous suffit de vous reporter au numéro de l'image de la banque comme d'habitude.

Pour afficher l'image inversée dans l'axe des abscisses (image du personnage marchant vers la gauche), vous forcez le bit 15 du numéro de l'image à 1. Ne paniquez-pas, vous pouvez simplement le faire avec:

```
$8000+Numéro d'image
```

Donc:

```
Bob 1,160,100,1
```

affiche votre personnage se dirigeant vers la droite et

```
Bob 1,160,100,$8000+1
```

l'affiche marchant vers la gauche. Le même principe est utilisé pour l'inversion verticale. Le bit 14 est utilisé. Ajouter $4000 au numéro de l'image. Pour obtenir une inversion verticale et horizontale, utilisez $C000.

La symétrie est complète: le POINT CHAUD (HOT SPOT) du bob est également inversé. Par exemple, si nous avions placé le point chaud sur X sous les pieds de notre personnage, il se trouverait également sous ses pieds dans la version inversée. Par conséquent, faites attention si vous fixez le point chaud dans le coin supérieur gauche

d'un bob, l'image inversée s'affichera en haut et à gauche!
Vous allez dire que l'utilisation de $8000 et $C000 est un peu étrange. Nous avons prévu des fonctions spéciales qui améliorent l'interface AMOS:

=HREV(image), ajoute $8000 à l'image
=HREV(image), ajoute $4000
=REV(image), ajoute $C000

Utilisez-les à la place des valeurs hex:

```
Bob 1,160,100,10
Bob 1,160,100,HREV(10)
Bob 1,160,100,VREV(10)
Bob 1,160,100,REV(10)
```

Pour faciliter l'utilisation de la bascule des bobs dans AMAL, nous avons mis en place une évaluation Hexadécimale. Par conséquent, vous pouvez utiliser la notation hex pour vous reporter aux bobs inversés. Si hex vous fait peur, ajoutez simplement $8000, $4000 ou $C000 avant toutes les références de vos chaînes AMAL. Exemple:

Première chaîne AMAL:

```
"Anim 0,(1,2)(2,2)(3,2)(4,2)"
```

Nouvelle chaîne inversée:

```
"Anim 0,($8000+1,2)($8000+2)($8000+3)($8000+4)"
```

ou...

```
"Anim 0,($8000+1,2)($8000+2,2)($8000+3,2)($8000+4,2)
```

Si vous utilisez un registre pour calculer le numéro de l'image, n'essayez-pas de modifier le calcul lui-même, seulement lorsque vous affectez le registre à l'image.
Première chaîne AMAL:

```
For R0=1 To 10: Let A=R0; Next R0
```

Nouvelle chaîne:

```
For R0=1 To 10; Let A=$C000+R0; Next R0
```

Comment la routine de bascule fonctionne-t-elle?

Il est vraiment important de comprendre comment cette routine fonctionne à l'intérieur de la machine pour que vous ne demandiez pas au système de faire des choses pour

lesquelles il n'est pas conçu.

Le système d'inversion est conçu pour libérer de la mémoire avant d'être rapide (bien que cela nous gênerait pas s'il était en fait plus rapide, n'est-ce-pas?). François a dû faire des concessions pour le rendre rapide, facile à utiliser et puissant.

La routine fonctionne en fait au beau milieu et n'utilise pas de mémoire supplémentaire. Les bobs sont basculés lors du processus d'actualisation, juste avant qu'un bob ne soit retracé à l'écran. AMOS cherche à voir si l'image doit être basculée dans la banque. Si c'est le cas, elle est basculée et un indicateur est activé dans la banque. A la prochaine actualisation, si l'image du bob n'a pas changé, elle ne sera pas rebasculée dans la banque afin d'économiser beaucoup de temps.

Si vous comprenez les explications ci-dessus, vous vous rendrez compte de l'existence d'un gros obstacle. Il n'est pas recommandé d'utiliser plus d'un bob basculé désignant la même image. Etudions l'exemple suivant:

```
Bob 1,160,100,1
Bob 2,160,150,$8001
Bob 3,20,20,$4001
Bob 4,20,100,$C001
Update
```

Lors du processus d'ACTUALISATION, AMOS saisira d'abord le bob#1. Pas de problème, il est dans la bonne position. Ensuite, le bob#2 - AMOS doit l'inverser en X. Le bob#3 a besoin d'inverser un X et un Y) pour remettre le bob dans sa position initiale en X!). Puis le bob#4 a besoin d'inverser un X.

A la prochaine actualisation, à condition que l'image du bob n'aie pas changé, pour afficher le bob#1, AMOS devra le basculer en X et en Y, puis procéder au bob#2...

Comme vous pouvez le constater, pour chaque ACTUALISATION, c'est-à- dire à chaque 50ème de seconde, si les bobs se déplacent, ils doivent être inversés! Cela marche mais demande beaucoup de temps du processeur et l'animation sera désastreuse.

Donc, la règle en or est d'utiliser les bobs inversés pour les objets seuls à l'écran (ou soyez sûr que les images normales et inversées ne sont pas affichées en même temps à l'écran). Si vous voulez, vous pouvez avoir deux bobs comme ceci, expérimentez!

Nous vous avons dit qu'auparavant, l'utilisation de ce système était destinée aux bobs. Le système est entièrement automatique avec les bobs. Mais puisqu'il affecte directement la banque de sprites, vous pouvez l'utiliser avec les sprites.

Lorsqu'un sprite machine est calculé, AMOS fait une recherche dans la banque de sprites et en saisit l'image. Si l'image est inversée à ce moment, le sprite machine affichera une image inversée. Vous pouvez alors avoir des sprites machines inversés en utilisant cette méthode. Mais vous ne pouvez pas faire ceci par exemple:

```
Sprite 1,200,200,$8001
```

Coller des bobs basculés

PASTE BOB accepte également des images inversées. Un simple truc pour inverser

une image dans la banque sans avoir à afficher un bob est de COLLER l'image inversée en dehors de l'écran. Exemple:

```
Paste Bob 500,500,$C000
```

Ceci inversera l'image 4 de la banque, sans aucune sortie à l'écran (et rapidement).

La détection de collision

C'est un point important et vous devez être très prudent lorsque vous détectez des collisions avec des bobs inversés!

La détection de collision utilise les formes des bobs de la banque au moment même où cette fonction est appelée. Etudions un exemple où la détection ne marche jamais:

```
Bob 1,160,100,1
Do
    Bob 2,XScreen(XMouse),YScreen(YMouse),$8001
    Wait Vbl
    Exit if Bob Col(1)
Loop
```

Pourquoi cela ne marche-t-il pas? Nous avons deux images inversées de même définition dans la banque. Après le processus d'actualisation, l'image de la banque est inversé à gauche. Par conséquent, l'instruction Bob Col saisira la forme du bob 1, l'image inversée et cela ne marchera pas!

Rappelez-vous donc ce que tonton François vous dit: Vous ne devez jamais utiliser la détection de collision avec plus d'une image inversée à l'écran!

Comment est-elle codée dans la banque de sprites?

Deux bits du point chaud x de chaque image sont utilisées pour indiquer le point de bascule. (Adresse SPRITE BASE + 6).

```
Bit#15 pour X: 0 si normal, 1 si inversé
Bit#14 pour Y: 0 si normal, 1 si inversé
```

Avant de DEMARRER (RUN) et de SAUVEGARDER, la banque est réinitialisée pour qu'elle soit toujours compatible à la version 1.1.

Bascule de blocs

La routine de bascule peut également être utilisée pour les blocs.

HREV BLOCK *(Basculer un bloc horizontalement)*

HREV BLOCK image

Bascule l'image du numéro du bloc horizontalement VREV BLOCK image (Basculer un bloc verticalement)
VREV BLOCK image

Bascule l'image du numéro du bloc verticalement

Comprimer une routine de bob

Cette routine fut initialement écrite pour un programme de Fun School 3 (L'Ecole des Malins 3) appelé LETTRES. Chaque lettre de l'alphabet était affichée comme un seul grand sprite. Il fallait donc 52 images différentes, ce qui demandait 110k de mémoire. Puisque deux images étaient affichées à la fois, la plupart de cet espace était en fait gâché. Nous avons donc demandé à François Lionet, notre génie de service, d'écrire une petite routine pour comprimer les sprites inutilisés dans une banque de mémoire de réserve. Ceci nous permettait de comprimer toute une banque de sprites en 26K!

AMOS SQUASH est maintenant disponible dans le domaine publique. N'hésitez pas à l'utiliser dans n'importe quel de vos propres programmes. Il est particulièrement utile pour les très nombreux niveaux d'un jeu d'arcade. Ils peuvent être comprimés en une fraction de leur taille normale et peuvent être retrouvés en un instant au point approprié de votre programme.

Utiliser AMOS Squash

Le programme est livré en deux parties. Le premier programme charge une banque de sprite en mémoire et comprime vos images. Vous le trouverez dans le fichier Squash_a_bob.AMOS.
Pour utiliser cette routine, il vous faut exécuter la simple procédure suivante:

- Chargez une banque de sprite du disque en utilisant le sélecteur de fichier.
- Entrez le numéro du premier sprite que vous souhaitez comprimer.
- Entrez le numéro du dernier sprite de votre liste.
- Entrez le nombre de couleurs utilisées par les images de votre sprite.
- Choisissez un nouvelle banque de mémoire pour contenir vos images comprimées.

La routine du squasher s'exécutera et vos images sélectionnées seront rapidement comprimées. Il vous sera ensuite donné l'option de sauvegarder vos images comprimées sur disque.

Notez qu'aucune information de couleurs n'est sauvegardée avec ces banques. Vous pouvez donc prendre note des paramètres courant de la couleur avant d'avancer plus loin.

Après avoir comprimé vos images, vous pouvez les charger dans vos programmes AMOS Basic à l'aide de trois petites procédures. Vous trouverez celles-ci dans le fichier Squash_procs.AMOS et vous pouvez les intégrer dans votre programme en utilisant la

commande FUSIONNER (MERGE) du menu de l'Editeur AMOS.

Les routines de décompression sont extrêmement faciles à utiliser. La première étape consiste à charger votre banque de sprites d'origine depuis le mode direct et d'effacer les images que vous venez de comprimer. Utilisez une ligne comme suit:

Del Sprite start To finish

Elle efface vos anciennes images de la mémoire et vous permet d'économiser une quantité considérable d'espace de rangement. Vous pouvez alors entrer vos images comprimées du disque à l'aide de la commande LOAD:

Load "images.abk"

Au début de votre programme, vous devez initialiser le système de compression en faisant un appel à la procédure PBOB_INIT.

PBOB_INIT[bank,cols,max_x,max_y]

Où:

bank: est le numéro de banque contenant les images comprimées

cols: contient le nombre de couleurs

max_x: mémorise la largeur maximale de vos images

max_y: mémorise la hauteur maximale de vos images

L'action de cette procédure est de préparer un écran temporaire pour l'utilitaire squasher. Vous pouvez décomprimer vos images lorsque vous le voulez, en faisant un simple appel à la routine PBOB.

PBOB[source,dest]

source est le numéro de l'image à décomprimer. C'est le numéro d'origine de l'image de la banque de sprite.

dest est le numéro de la nouvelle image que vous souhaitez installer dans la banque de sprite. Choisissez le numéro du dernier sprite de la banque et ajoutez un.

Après avoir installé votre image en mémoire, vous pouvez l'animer directement à l'aide des diverses commandes SPRITE ou BOB. PBOB peut être appelé autant de fois que vous voulez. Le même numéro d'image peut donc être utilisé et réutilisé.

Finalement, faites un appel à la procédure PBOB_END vers la fin de votre programme. Ceci effacera l'écran caché créé par PBOB_INIT.

13 Contrôle de l'objet

Vous allez découvrir au cours de ce chapitre comment les divers objets générés au moyen des commandes sprites et des commandes bob peuvent être contrôlés par un programme AMOS Basic. Les sujets à l'étude comprennent la détection de collision, l'utilisation de la souris et la lecture du joystick.

La flèche de la souris

Pour le programmateur de jeu, la flèche de la souris est une alternative précieuse au joystick habituel. Grâce à la commande CHANGE MOUSE, vous pouvez remplacer la souris par une image de la banque sprite courante. Il existe également un bloc d'instructions vous permettant de déterminer la position et l'état de la souris à n'importe quel moment. Il englobe les instructions X MOUSE, Y MOUSE et MOUSE KEY.

HIDE *(Supprimer la flèche de la souris de l'écran)*

HIDE [ON]

Cette commande supprime complètement la flèche de la souris de l'écran. Le système mémorise le nombre d'appels de cette fonction. Il doit être égal au nombre d'instructions SHOW pour que la flèche revienne à l'écran.

Il existe une variante de cette instruction qui est accessible avec HIDE ON. Elle ignore le nombre d'appels et cache **toujours** la souris, peu importe le nombre d'appels à la commande SHOW.

Notez que HIDE rend seulement la flèche de la souris invisible. Il n'a aucun effet sur d'autres commandes AMOS; vous pouvez donc continuer, comme d'habitude, à utiliser les fonctions X MOUSE et Y MOUSE pour lire les coordonnées de la souris.

SHOW *(Activer la flèche de la souris)*

SHOW [ON]

Cette commande renvoie la flèche de la souris à l'écran après l'exécution d'une instruction HIDE. Le nombre de commandes HIDE précédentes est conservé par défaut. Si le nombre de SHOW est inférieur au nombre de HIDE, la flèche reste invisible. Pour désactiver le compteur et activer immédiatement la souris, utilisez plutôt la commande SHOW ON.

CHANGE MOUSE *(Changer la forme de la flèche de la souris)*

CHANGE MOUSE m

Cette commande vous permet de changer la forme de la souris à votre gré. Trois motifs standard de souris vous sont proposés. Vous pouvez les affecter en choisissant un nombre compris entre 1 et 3 comme suit:

M	**Forme**
1	Flèche (Défaut)
2	Viseur
3	Horloge

Si vous donnez une valeur à *m* supérieure à 3, m est supposé se rapporter à une image conservée dans la banque sprite. Le numéro de cette image est déterminé par l'expression *I=m-3*. Par conséquent, l'image numéro 1 serait mise en place par une valeur de 4 et l'image numéro 2 recevrait une valeur de 5.

Pour utiliser cette option, la largeur de votre image de sprites doit être exactement de 16 pixels et doit comporter au plus quatre couleurs. Cependant, aucune limite n'est imposée à la hauteur de votre image. Voir également à HIDE, SHOW, X MOUSE, Y MOUSE, MOUSE KEY, LIMIT MOUSE.

MOUSE KEY *(Lire l'état des boutons de la souris)*

k=MOUSE KEY

La fonction MOUSE KEY vous permet de vérifier rapidement si vous avez appuyé sur un ou plusieurs boutons de la souris. Elle renvoie une configuration binaire contenant l'état courant des boutons de la souris.

La syntaxe de cette configuration binaire est la suivante:

Bit 0	Forcé à 1 si le bouton GAUCHE est enfoncé, sinon zéro
Bit 1	Etat du bouton DROIT en utilisant la même syntaxe
Bit 2	Etat du TROISIEME bouton s'il existe

Exemple:

```
Curs Off
Do
   Locate 0,0
   M=Mouse Key:Print "Configuration binaire";Bin$(M,8);" Number",M
Loop
```

= MOUSE CLICK *(Vérifier le déclic de la souris)*

c=MOUSE CLICK

La fonction MOUSE CLICK vérifie que l'utilisateur a "cliqué" sur un bouton de la souris. Elle renvoie une configuration binaire de syntaxe:

Bit 0	Test d'un seul passage pour le bouton GAUCHE
Bit 1	Test d'un seul passage pour le bouton DROIT
Bit 2	Test d'un seul passage pour le TROISIEME bouton s'il existe

Les tests d'un seul passage sont seulement forcés à 1 si le bouton de la souris vient d'être abaissé. Ces bits sont automatiquement remis à zéro après qu'ils aient subi un seul test. Ces tests vérifieront donc l'appui d'un bouton à la fois. Voici un exemple:

```
Curs Off
Do
   M=Mouse Key
   If M<>0 Then Print "Configuration binaire";Bin$(M,8);" Number",M
Loop
```

=X MOUSE= *(Obtenir/déterminer l'abscisse X de la flèche de la souris)*

x1=X MOUSE

X MOUSE renvoie l'abscisse courante X de la flèche de la souris en notation machine.

Vous pouvez également utiliser cette fonction **pour** déplacer la souris vers un point précis de l'écran. Pour cela, il vous suffit d'affecter X MOUSE d'une valeur comme pour une variable Basic. Par exemple:

```
X Mouse=150 : Rem Déplace la souris vers la coordonnée machine 150, ord. Y n'est pas
              affectée
```

=Y MOUSE= *(Obtenir/déterminer l'ordonnée de la flèche de la souris)*

y1=Y MOUSE

Cette fonction renvoie l'ordonnée Y de la flèche de la souris. Comme avec Y MOUSE, le résultat est exprimé en coordonnées *machine* et non en coordonnées écran plus habituelles. Vous pouvez également utiliser Y MOUSE pour repositionner la flèche de la souris à l'écran. Il vous suffit de charger la nouvelle coordonnée dans Y MOUSE de la façon suivante:

```
Y Mouse=100
```

Pour obtenir une illustration des fonctions X MOUSE et Y MOUSE, chargez **le fichier EXEMPLE 13.1** du dossier MANUEL.

LIMIT MOUSE *(Renfermer la souris dans une portion de l'écran)*

LIMIT MOUSE x1,y1 TO x2,y2

Cette commande renferme les déplacements de la souris dans une zone rectangulaire définie par les coordonnées machine *(x1,y1)* et *(x2,y2)*. *x1,y1* marque l'angle supérieur gauche de ce cadre et *x2,y2* le point diagonalement opposé. Notez qu'à la différence de LIMIT BOB, la souris est complètement enfermée dans cette zone et ne peut pas être déplacée au-delà de son cadre. Chargez **le fichier EXEMPLE 13.2** du dossier MANUEL pour obtenir une démonstration de cette commande. Il vous suffit d'utiliser l'instruction suivante sans paramètre pour replacer la souris dans l'intégralité de la zone écran.

Limit Mouse

La lecture du joystick

AMOS Basic offre six fonctions vous permettant de vérifier immédiatement les déplacements d'un joystick que vous avez préalablement branché.

= JOY *(Lire le joystick)*

d=JOY(j)

Cette fonction renvoie un nombre binaire représentant l'état courant d'un joystick au port numéro *j*. Normalement, votre joystick est branché dans la prise gauche (numéro 1). Toutefois, vous pouvez débrancher la souris de la prise de droite et la remplacer par un joystick. Celui-ci est accessible en utilisant le port numéro zéro.

Vous pouvez lire l'état du joystick en examinant la valeur des bits binaires. Chaque bit indique une action précise effectuée par l'utilisateur. Si un bit est forcé à un, alors le test se révèle être positif, signifiant que le joystick a été déplacé dans la direction voulue.

Voici une liste des divers bits et leur signification:

Numéro de bit	Mouvement
0	Joystick déplacé vers le haut
1	Joystick déplacé vers le bas
2	Joystick déplacé vers la gauche
3	Joystick déplacé vers la droite
4	Bouton de tir actionné

Ne vous inquiétez pas si vous n'êtes pas très familier avec la notation binaire. Vous pouvez également accéder à chacune des directions au moyen des fonctions JLEFT, JRIGHT, JUP, JDOWN et FIRE. Référez-vous au fichier **EXEMPLE 13.3** pour une démonstration de cette commande.

=JLEFT *(Tester le déplacement du joystick vers la gauche)*

x=JLEFT(j)

Cette commande renvoie une valeur de -1(Vrai) si le joystick du port *j* a été manoeuvré

vers la gauche, sinon une valeur de 0 (Faux) est renvoyée. Les prises du joystick sont numérotées de droite à gauche à partir de zéro. Donc, vous pouvez accéder à la prise par défaut du joystick en utilisant le port numéro 1. Par exemple:

```
Do
   If Jleft(1) Then Print "GAUCHE"
Loop
```

=JRIGHT *(Tester le déplacement du joystick vers la droite)*

x=JRIGHT(j)

Cette commande teste le joystick numéro j et renvoie la valeur -1(Vrai) si le joystick a été manoeuvré vers la droite, sinon la valeur zéro est retournée (Faux). Voir JLEFT, JUP, JDOWN.

=JUP *(Tester le déplacement du joystick vers le haut)*

x=JUP(j)

Cette commande renvoie la valeur -1 si le joystick j a été poussé vers le haut, sinon la valeur zéro est retournée. Voir JRIGHT, JLEFT, JDOWN.

=JDOWN *(Tester le déplacement du joystick vers le bas)*

x=JDOWN(j)

La fonction JDOWN renvoie la valeur -1 si le joystick a été abaissé, sinon la valeur zéro est retournée. Voir JRIGHT, JLEFT, JUP.

=FIRE *(Tester l'état du bouton de tir)*

x=FIRE(j)

Cette fonction renvoie seulement une valeur de -1 si le bouton de tir du joystick numéro j a été abaissé. Voir JUP, JDOWN, JLEFT, JRIGHT, JOY.

Détecter les collisions

Si vous écrivez un jeu d'arcade, il vous faut absolument faire une recherche précise des collisions entre les divers objets à l'écran. AMOS Basic vous offre cinq puissantes fonctions permettant d'effectuer ces tests rapidement et facilement.

Détecter les collisions avec un sprite

SPRITE COL *(Détecter les collisions entre deux sprites machine)*

c=SPRITE COL(n[,s TO e])

Cette commande vous permet de tester d'une façon simple si deux ou plusieurs sprites se sont rencontrés à l'écran. Le numéro *n* se rapporte à un sprite machine actif dont on doit rechercher l'entrée en collision. Si une collision s'est produite, une valeur de -1 (vrai) est renvoyée, sinon le résultat est forcé à 0 (faux).

Cette fonction de forme standard recherche toutes les collisions. Mais vous pouvez également tester un groupe entier de sprites en utilisant une version prolongée de cette commande:

c=SPRITE COL n,s TO e

L'instruction ci-dessus recherche les collisions entre le sprite *n* et les sprites s à e (compris). Lorsque vous avez détecté une collision, vous pouvez alors obtenir les numéros des sprites qui se sont rencontrés au moyen de la fonction COL.

Notez qu'il vous faut d'abord créer un *masque* de *sprites* avec la commande MASK pour utiliser cette fonction, sinon vos collisions ne seront pas détectées. Vous trouverez un exemple détaillé de cette commande dans **le fichier EXEMPLE 13.4.**

Détecter les collisions avec un bob

BOB COL *(Détecter les collisions entre deux objets graphiques à manipuler)*

c=BOB(n [,s TO e])

La fonction BOB COL recherche la collision entre le bob numéro n et un autre bob. Si une collision est détectée, la valeur renvoyée en c est forcée à -1 (vrai), sinon elle est initialisée à 0 (faux).

La commande recherche normalement toutes les collisions mais vous pouvez préciser le groupe de bobs que vous voulez tester au moyen des paramètres optionnels de s à e. Vous pouvez examiner l'état de chacun de ces bobs au moyen de la commande COL. Voir **le fichier EXEMPLE 13.5.**

Collisions entre les bobs et les sprites

En AMOS Basic, vous n'êtes pas simplement limité à la détection de collisions entre les mêmes types d'objets. Il vous est également possible de rechercher les combinaisons de sprites et de bobs à l'écran.

SPRITEBOB COL *(Tester une collision entre les sprites et les bobs)*

c=SPRITEBOB COL (n [,s TO e])

Cette fonction recherche une collision entre le **sprite** *n* et un ou plusieurs **bobs**. La valeur de *c* est de -1 si une collision est localisée ou de 0 si il n'y a pas de collisions. Le premier et le dernier numéro spécifient la zone de détection des collisions entre les bobs

s et *e*. Si cette zone n'est pas délimitée, alors cette fonction testera tous les bobs actifs. Voir la commande COL pour plus de précisions.

Attention! La détection de collision entre un sprite et un bob est seulement possible sur un écran de basse résolution. En mode haute résolution, les dimensions en pixel choisies pour les bobs et les sprites sont tout à fait différentes et les résultats donnés par cette fonction sont faux.

BOBSPRITE COL *(Tester une collision entre les bobs et les sprites)*

c=BOBSPRITE COL (n,{s TO e])

La fonction BOB SPRITE COL recherche les collisions entre un seul bob et plusieurs sprites. Si le test se révèle positif, la commande renvoie une valeur de -1, sinon la valeur de 0 est renvoyée. Les paramètres optionnels liste un groupe de sprites devant être testés de s à e. Si vous omettez de préciser ce groupe, cette fonction recherchera tous les sprites couramment actifs.

Notez que BOB SPRITE COL fonctionne seulement sur les écrans à basse résolution. N'essayez donc pas de l'utiliser avec un mode à haute résolution ou vous obtiendrez des résultats imprévisibles. Vous pouvez trouver une démonstration complète de cette commande dans **le fichier EXEMPLE 13.6** du dossier MANUEL.

=COL (Tester l'état d'un sprite ou d'un bob après l'exécution d'une instruction de détection de collision)

c=COL(n)

Le tableau COL contient l'état de tous les objets qui ont été préalablement testés par les fonctions de détection de collision.

Chaque objet que vous avez recherché est associé à un seul élément de cette matrice. Cet élément reçoit une valeur de -1 si une collision est détectée avec l'objet numéro *n* ou de 0 dans le cas contraire. Le système de numérotation est simple: le premier élément de la matrice représente l'état de l'objet numéro 1, le deuxième l'objet numéro 2, etc. Référez-vous au fichier **EXEMPLE 13.7.**

Si vous utilisez les instructions SPRITE COL ou BOB SPRITE COL, les objets sont des sprites machine, sinon ils sont des BOBs. Afin d'éviter toute confusion, il est bon d'appeler cette instruction immédiatement après avoir exécuté la commande de détection.

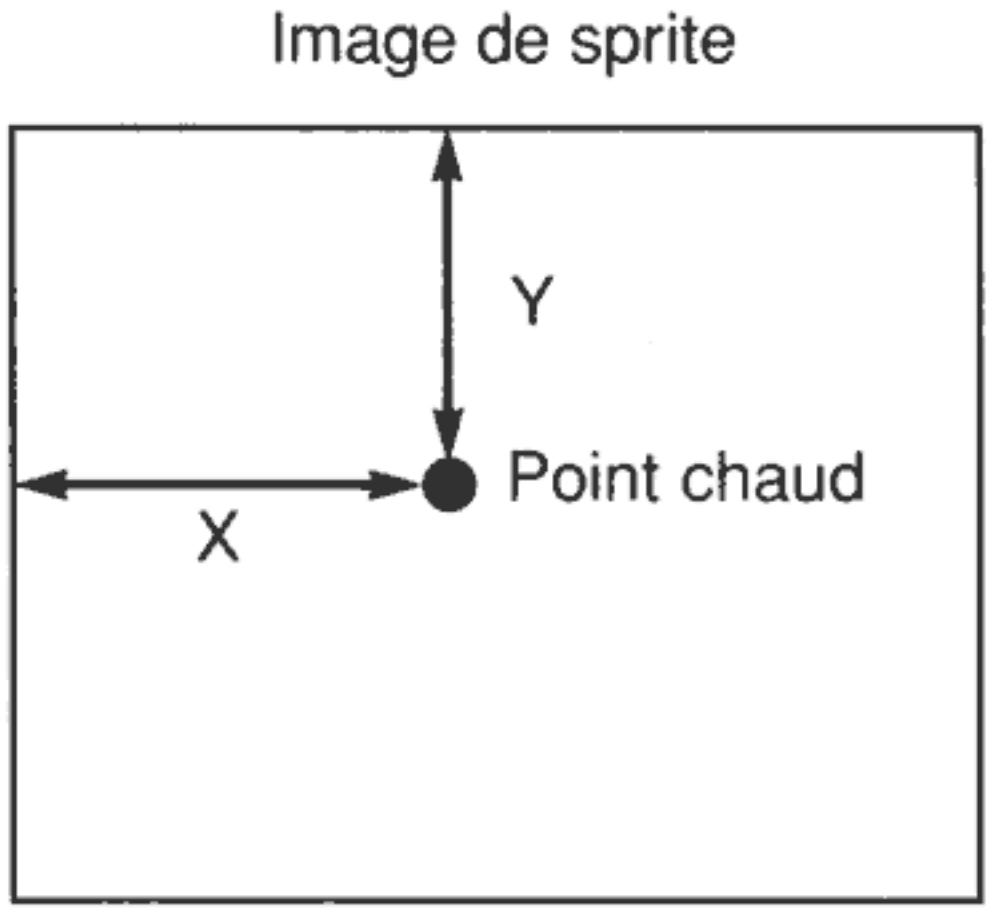
HOT SPOT *(Déterminer le point chaud d'une image dans la banque sprite)*

HOT SPOT image,x,y
HOT SPOT image,p

Cette commande détermine le point chaud d'une image conservée dans la banque sprite courante. Le point chaud de l'objet est utilisée comme point de référence pour tous les calculs de coordonnées. Il existe deux variantes de cette instruction.

HOT SPOT image,x,y

Les coordonnées *x* et *y* sont déterminées à partir de l'angle supérieur gauche de l'image. Elles sont ajoutées aux coordonnées du sprite ou du bob pour positionner précisément un objet à l'écran.



Notez que le point chaud peut très bien se trouver en dehors de l'image réelle.

HOT SPOT image.p

C'est une forme abrégée de l'instruction qui déplace le point chaud vers une des neuf positions prédéfinies. Les positions sont affichées dans le tableau ci-dessous dans lequel la valeur de $11 représente le point central de l'image.

$00	$10	$20
$01	$11	$21
$02	$12	$22

Référez-vous au fichier **EXEMPLE 13.8.**

MAKE MASK *(Créer un masque autour d'une image pour la détection de collision)*

MAKE MASK [n]

Cette commande définit un masque autour de l'image numéro *n* de la banque sprite. Toutes les commandes de détection de collision AMOS Basic l'utilisent. Vous devez donc créer un masque pour chaque objet que vous souhaitez recherchez. Si vous

omettez de préciser le numéro *n* de l'image, un masque sera créé pour chaque image de la banque sprite. Cela peut prendre un peu de temps.

Il est important de noter que les masques sont automatiquement produits lors du premier tracé d'un bob à l'écran. Cela pourrait entraîner un retard important dans le déroulement de votre programme, donc il est bon de faire un appel explicite à cette commande lors de votre procédure d'initialisation.

Entrées en collision avec les zones rectangulaires

AMOS Basic vous offre un certain nombre de fonctions vous permettant de vérifier si un sprite ou un bob a pénétré une zone rectangulaire de l'écran.

Ces *zones rectangulaires* sont particulièrement utiles pour la détection de collision dans les jeux à effets de rebond tels que Arkanoid puisque chaque bloc peut être affecté de sa propre zone d'écran. Vous pouvez également utiliser ces zones pour élaborer les boutons ainsi que les interrupteurs des panneaux de commandes et des boîtes de dialogue.

RESERVE ZONE *(Réserver de l'espace mémoire pour la définition d'une zone de détection)*

RESERVE ZONE [n]

Cette commande affecte suffisamment de mémoire pour n zones de détection. Elle doit toujours être utilisée avant de définir une zone avec SET ZONE.

La seule limite imposée au nombre de zones est la quantité de mémoire disponible. Il est donc parfaitement possible de définir des centaines ou même des milliers de zones dans un de vos programmes.

Pour effacer les définitions courantes de la zone et réaffecter la mémoire au programme principal, il vous suffit de taper RESERVE ZONE sans paramètre.

SET ZONE *(Définir une zone de test)*

SET ZONE z,x1,y1 TO x2,y2

Cette commande définit une zone rectangulaire pouvant être testée ultérieurement en utilisant les diverses commandes ZONE. *z* indique le numéro de la zone devant être créée et les coordonnées *x1,y1* et *x2,y2* marquent la position de l'angle supérieur gauche et de l'angle inférieur droit du rectangle.

Avant d'utiliser cette instruction, il vous faudra réserver de la mémoire pour la définition de vos zones au moyen de RESERVE ZONE.

Voir ZONE, RESET ZONE, RESERVE ZONE, ZONE$.

=ZONE *(Renvoyer la zone aux coordonnées écran demandées)*

t-ZONE([x],x,y)

Cette commande renvoie le numéro de la zone écran aux coordonnées graphiques *x,y*. Les coordonnées se rapportent normalement à l'écran courant; vous pouvez également intégrer un écran optionnel numéro s à cette fonction.

Après l'appel de ZONE, t représente le numéro de la zone aux coordonnées spécifiées ou une valeur de 0 (faux).

Notez que ZONE renvoie seulement la première zone au point de ces coordonnées; elle ne détectera aucune autre zone se trouvant à l'intérieur de cette première.

Afin de démontrer cette commande, nous avons joint au dossier MANUEL deux programmes à titre d'exemple. Vous pouvez les trouver dans **les fichiers EXEMPLE 13.9** ET **EXEMPLE 13.10**. N'hésitez pas à les modifier et à les intégrer dans vos propres jeux.

Il est possible d'utiliser cette fonction en combinaison avec les fonctions X BOB et Y BOB pour détecter si un bob a pénétré une zone spécifique de l'écran. A cette fin, utilisez le code suivant:

```
X=Zone(X Bob(n),Y Bob(n))
```

Voir HZONE, SET ZONE, RESET ZONE, X BOB, Y BOB.

=ZONE *(Renvoyer la zone aux coordonnées machine demandées)*

t=HZONE ([x],x,y)

Cette commande est presque identique à ZONE à l'exception que la position de l'écran est maintenant mesurée en coordonnées machine. Vous pouvez donc utiliser cette fonction pour détecter le moment où un sprite machine pénétre une de vos zones d'écran. Par exemple:

```
X=Hzone(X Sprite(n),Y Sprite(n))
```

Vous trouverez une démonstration de cette commande dans le fichier EXEMPLE 13.11.
Voir ZONE, MOUSE ZONE, SET ZONE, RESERVE ZONE, ZONE$.

=MOUSE ZONE *(Vérifier si la flèche de la souris a pénétré une zone)*

x=MOUSE ZONE

Cette fonction renvoie le numéro de la zone d'écran actuellement occupée par la flèche de la souris. Elle est équivalente à la ligne:

```
H=Hzone(X mouse, Y mouse)
```

Voir ZONE, HZONE, SET ZONE, ZONE$.

RESET ZONE *(Effacer une zone)*

RESET ZONE [z]

Cette commande désactive d'une façon permanente les zones créées par SET ZONE. Si la zone optionnelle numéro *z* se trouve parmi celles-ci, alors seule cette zone sera réinitialisée, sinon toutes les zones seront affectées. Notez que RESET ZONE efface seulement les définitions de zone, elle ne renvoie pas la mémoire attribuée par RESERVE ZONE.

Valeurs de priorité du bob

PRIORITY ON/OFF *(Changer le mode de priorité)*

PRIORITE ON/OFF

Chaque bob est affecté d'une valeur de *priorité* comprise entre 0 et 63. AMOS Basic utilise ce numéro pour décider de l'ordre dans lequel les objets seront affichés à l'écran. En règle générale, les bobs ayant le rang le plus haut s'affichent toujours devant les objets de rang inférieur. La valeur de priorité dépend du numéro attribué au bob.

Vous devez vous rappeler de cette règle lorsque vous affectez un numéro à un bob. L'apparence de vos objets à l'écran dépend du choix du numéro; les effets obtenus peuvent être donc très différents.

En complément du système standard, il est également possible de disposer les bobs selon leur position à l'écran. PRIORITY ON affecte les bobs des valeurs de priorité les plus grandes en leur donnant une ordonnée Y la plus élevée. Cette commande vous permet de créer des effets de perspective dans vos jeux. Examinez l'exemple ci-dessous:

```
Load "AMOS_DATA:Sprites/Monkey_right.abk" : Cls : Flash Off : Get Sprite Palette
Double Buffer
Priority Off : Rem Définir Le Mode Normal
Bob 1,160,100,2 : Bob 2,0,74,2 : Bob 3,320,128,2
Channel 2 To Bob 2 : Channel 3 To Bob 3
Amal 2," Loop: M 320,0,320 ; M -320,0,320 ; Jump Loop"
Amal 3," Loop: M -320,0,320 ; M 320,0,320 ; Jump Loop"
Amal On
Wait Key
Priority On : Rem Définir Le Mode Y
Wait Key
```

Les deux bobs actifs passent dessous l'objet du milieu. Si vous changez le système de priorité en faisant appel à PRIORITE ON, les bobs se rangent dans l'ordre croissant de leur ordonnée. Donc, le bob trois se déplace au-dessus du bob un tandis que le bob deux passe lentement derrière ce dernier.

CONSEIL: En général, il est préférable de positionner le point chaud du sprite à sa base parce que l'ordonnée Y utilisée par cette commande se rapporte à la position du point chaud à l'écran. Remarquez également que vous pouvez utiliser l'instruction PRIORITY OFF pour réinitialiser la valeur de priorité.

PRIORITY REVERSE ON/OFF *(Changer l'ordre de dessin des bobs à l'écran)*

PRIORITY REVERSE ON
PRIORITY REVERSE OFF

PRIORITY REVERSE ON inverse toute la table de priorité des bobs. Cela signifie que le bob numéro 1 se retrouve premier devant tous les autres bobs, le bob 2 viendra second etc... Cette liste de priorité est compatible avec le STOS.

Cette instruction présente une autre fonction vraiment agréable lorsqu'elle est utilisée avec PRIORITY ON. Les bobs ne sont plus triés de HAUT en BAS! Le bob le plus haut à l'écran sera affiché devant les autres.

Diverses commandes

UPDATE *(Modifier les actualisations automatiques des sprites ou des bobs)*

UPDATE [ON/OFF]

En général, les bobs que vous tracez à l'écran s'affichent automatiquement à leur animation ou à leur déplacement. Vous pouvez temporairement désactiver cette fonction au moyen de la commande UPDATE OFF. Si les actualisations ne sont pas actives, les commandes SPRITE, BOB et AMAL n'auront apparemment pas d'effets. En fait, toutes vos animations s'effectueront correctement mais seulement les résultats ne s'afficheront pas à l'écran. Vous pouvez faire exécuter cette opération de retracé à votre convenance par la commande UPDATE. Voici trois différentes syntaxes de l'instruction UPDATE:

UPDATE OFF

Désactive l'actualisation automatique des bobs et des sprites. Les déplacements ou les animations paraîtront figées.

UPDATE

Retrace les sprites ayant changé leur position d'origine.

UPDATE ON

Remet l'actualisation de sprite sur normal. Référez-vous au fichier **EXEMPLE 13.12.** Voir UPDATE EVERY, SYNCHRO, SPRITE UPDATE, BOB UPDATE.

14 AMAL

Si vous souhaitez produire le déplacement graduel que l'on trouve dans un jeu d'arcade, il vous faut déplacer chaque objet à l'écran des douzaines de fois par seconde. C'est vraiment difficile même en code machine et c'est bien au-dessous des capacités de la version la plus rapide du Basic.

AMOS contourne ce problème par l'intégration d'un langage puissant d'animation qui est exécuté indépendamment de vos programmes Basic. Ce langage est capable de produire des effets à grande vitesse d'animation qui seraient impossibles en Basic normal.

L'**A**MOS **A**nimation **L**anguage (AMAL) est unique à l'AMOS Basic. Vous pouvez vous en servir pour animer tout ce que vous voulez, d'un sprite à l'écran entier à une vitesse incroyable. Vous pouvez exécuter 16 programmes AMAL simultanément en utilisant des interruptions.

Chaque programme commande les déplacements d'un seul objet à l'écran. Vous pouvez déplacer les objets selon des schémas d'attaque pré- définis, créés à partir d'un accessoire externe de l'éditeur. Vous pouvez également les commander directement depuis la souris ou le joystick.

Il faut tester la polyvalence du système AMAL pour y croire. Chargez 1 du dossier MANUEL pour une démonstration complète.

Les principes d'AMAL

AMAL est en réalité une version simplifiée du Basic qui a été particulièrement optimisé pour obtenir la plus grande possible vitesse. Comme avec le Basic, vous disposez d'instructions pour le contrôle du programme (Jump), de prises de décision (If) et de blocs de répétition de code sous le format de boucles (For...Next). La force réelle du langage est révélée lors du déroulement du programme AMAL. Non seulement les commandes s'exécutent très rapidement mais tous les programmes AMAL sont **compilés** avant leur phase de déroulement.

Les commandes AMAL sont entrées au moyen de courts mots de passe consistant en une ou plusieurs lettres majuscules. Les caractères minuscules ne sont pas reconnus. Ceci vous permet de rendre vos instructions AMAL plus lisibles. Donc, vous pouvez entrer la commande M pour Move et L pour Let.

Les instructions AMAL peuvent être séparées par tous les caractères virtuellement inutilisés, y compris les espaces. Toutefois, vous ne pouvez pas utiliser les deux points ":" à cette fin, puisqu'il sert à définir une étiquette. Nous vous conseillons d'utiliser un point- virgule ";" pour séparer les commandes et vous éviter ainsi d'éventuels maux de tête AMAL.

Il existe deux façons de créer des programmes AMAL. La première est de produire les séquences d'animation avec le programme accessoire AMAL et les sauvegarder dans une banque de mémoire ou bien vous définissez vos animations à l'intérieur de l'AMOS Basic au moyen des commandes AMAL. La syntaxe générale de cette instruction est la suivante:

```
AMAL n,a$
```

n est le numéro d'identification de votre nouveau programme AMAL. Par défaut, tous les programmes sont affectés au sprite machine approprié. Donc, le premier programme AMAL commande le sprite numéro un, le deuxième sprite numéro deux et ainsi de suite. Vous pouvez changer cette sélection à votre gré au moyen d'une commande individuelle CHANNEL. *a$* est une chaîne contenant une liste d'instructions AMAL devant être exécutées dans votre programme. Voici un exemple simple:

```
Load "AMOS_DATA:Sprites/Monkey_right.abk" : Rem Charger des sprites modèles
Get Sprite Palette
Sprite 8,130,150,1 : Rem Place un sprite à l'écran
Amal 8, "S: M 300,200,100 ; M -300,-200,100 J S" : Rem Définir un petit programme AMAL
Amal On 8 : Rem Activer le programme AMAL numéro huit
Direct
```

Le programme vous redonne directement la main en mode direct au moyen de la commande DIRECT. Essayez à ce moment-là de taper quelques commandes Basic . Vous constaterez alors que la séquence de déplacement se poursuit sans perturber le reste du système AMOS. Notez également que nous avons choisi le sprite 8 pour forcer l'utilisation d'un sprite calculé. Tous les sprites calculés compris entre 8 et 15 sont automatiquement affectés au numéro de canal équivalent par le système AMAL. Les procédures spéciales d'initialisation ne sont donc pas nécessaires. A moins que vous souhaitiez restreindre le nombre de sprites machine, il est plus sûr de se limiter aux sprites calculés dans vos programmes. Notez que nous avons activé le programme AMAL au moyen de la commande AMAL ON. La syntaxe de cette commande est la suivante:

AMAL ON[prog]

prog est le numéro du seul programme AMAL que vous souhaitez démarrer. Si vous omettez de spécifier ce numéro, tous vos programmes AMAL seront exécutés en même temps.

Guide d'initiation à l'AMAL

Nous allons maintenant vous offrir une visite guidée du système AMAL. Elle vous permettra de vous familiariser lentement avec les mécanismes des programmes AMAL, sans que vous ayez crainte d'être submergé de détails techniques.

Pour le moment, nous allons nous concentrer sur les déplacements des sprites, mais les mêmes principes peuvent également s'appliquer aux animations de bob ou aux animations d'écran.

Commencez par charger des sprites modèles en mémoire. Vous pouvez les trouver dans le dossier Sprites de la disquette de donnée AMOS. Pour obtenir un répertoire des fichiers Sprite, tapez l'instruction suivante depuis la fenêtre directe:

```
Dir "AMOS_DATA:"
```

Pour charger un fichier sprite, tapez la ligne:

```
Load "AMOS-DATA:Sprites/octopus.abk"
```

Déplacer un objet

Comme vous l'escomptiez d'un langage dédié à l'animation, AMAL vous permet de déplacer vos objets de bien différentes façons. La plus simple d'entre elles comprend l'utilisation de la commande Move.

Move *(Déplacer un objet)*

M w,h,n

La commande M déplace un objet de w unités vers la droite et de *h* unités vers le bas en exactement *n* pas. Si les coordonnées de votre objet sont (X,Y), alors l'objet se déplace progressivement vers la coordonnée x+W,Y+H.

Supposez que vous ayez un sprite aux coordonnées 100,100. L'instruction M 100,100,100 le déplacerait vers la coordonnée 200,200. La vitesse de ce déplacement dépend du nombre de pas. Plus *n* est élevé, plus le déplacement de chaque sprite sera court et le sprite avancera donc très lentement. Inversement, une petite valeur de *n* donne des grands pas de déplacement qui font brusquement avancer le sprite à grande vitesse au travers de l'écran. Voici quelques exemples de la commande Move:

```
Rem Ceci fait descendre un sprite octopus au moyen du AMAL
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,300,0,1
Amal 8, "M 0,250,50" : Amal On 8 : Wait Key

Rem Cette version fait traverser l'écran à un sprite octopus
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 9,150,150,1
Amal 9,"M 300,0,50" : Amal On 9 : Wait Key

Rem Fait descendre et traverser l'écran à l'octopus
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 10,150,150,1
Amal 10, "M 300,-100,50" : Amal On 10 : Wait Key

Rem Donne une démonstration des multiples commandes Move
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
M$="Move 300,0,50 ; Move -300,0,50"
Sprite 11,150,150,1
Amal 11,M$ : Amal On 11 : Wait Key
```

Remarquez la façon dont nous avons prolongé le M à Move dans le programme ci-dessus. Puisque les lettres "ove" sont en caractères minuscules, le système AMAL ne les reconnaît pas.

Au premier abord, Move est une petite instruction puissante mais pas bien

sensationnelle. Elle est idéale pour déplacer les objets comme les missiles, mais elle est peu inspirante.

En fait, c'est complètement faux parce que les paramètres de l'instruction ci-dessus ne sont pas limités à de simples nombres. Vous pouvez également utiliser des expressions arithmétiques complexes et y intégrer une des diverses fonctions AMAL bien utiles. Exemple:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette: Sprite 12, 150,150,1
Amal 12, "Move XM-X,YM-Y,32"
Amal On 12 : Wait Key
```

Cette instruction déplace graduellement le sprite calculé 12 vers la position courante de la souris. X et Y représentent les coordonnées de votre sprite et XM et YM sont les fonctions renvoyant les coordonnées courantes de la souris.

Il est possible d'exploiter cet effet dans des jeux comme Pac-Man pour faire **chasser** le personnage du joueur par vos objets. Vous trouverez une démonstration dans **2**.

Vous pouvez également utiliser la commande Move pour animer tout un écran. Voici un exemple simple:

```
Load Iff "AMOS_DATA;IFF/AMOSPIC.IFF",1
Channel 1 To Screen Display 1 : Rem Affecte le programme 1 à l'écran 1
Amal 1, "Move 0,-200,50 ; Move 0,200,50"
Amal On 1 : Direct
```

CHANNEL affecte un programme AMOS à un objet particulier. Nous étudierons cette commande en détail un plus tard. En attendant, voici sa syntaxe de base:

CHANNEL p To object n

p est le numéro de votre programme AMAL. Les valeurs autorisées sont comprises entre 0 et 63, bien que seuls les 16 premiers de ces programmes puissent être exécutés en utilisant des interruptions.

objet spécifie le type d'objet que vous souhaitez contrôler avec votre programme AMAL. Il est indiqué au moyen de l'une des instructions suivantes:

Sprite	(Valeurs supérieures à sept se rapportent aux sprites calculés)
Bob	(Objet de manipulation)
Screen-Display	(Utilisée pour déplacer l'image)
Screen Offset	(Scrolling machine)
Screen Size	(Change la taille de l'écran au moyen d'interruptions)
Rainbow	(Anime un arc-en-ciel)

n est le numéro de l'objet à animer. Cet objet doit être défini par la suite au moyen des instructions SPRITE, BOB ou SCREEN OPEN. Exemples:

```
Channel 2 To Bob 1 : Rem Animer Bob 1 au moyen du programme AMAL numéro 2
```

```
Channel 3 To Sprite 8 : Rem Affecter le canal trois à un sprite calculé
Channel 4 To Screen Display 0 : Rem Déplacer l'écran implicite via AMAL
Channel 5 To Screen Offset 0 : Rem Changer le déplacement d'écran dans l'AMAL
```

Animation

Anim *(Animer un objet)*

A n,(image,durée)(image,durée)...

L'instruction Anim fait passer un objet d'une séquence d'images à l'autre, en produisant un effet d'animation graduel. *n* représente le nombre de répétitions du cycle d'animation. Une valeur de zéro attribuée à ce paramètre permettra une animation en continu.

image spécifie le numéro d'image utilisée pour chaque phase de l'animation.

durée détermine la durée d'affichage de cette image à l'écran, mesurée en termes de 50ème de seconde. Exemples:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,260,100,1
Amal 8, "A 0,(1,2)(2,2)(3,2)(4,2)"
Amal On 8 : Direct

Load "AMOS_DATA:Sprites/Monkey_right.abk" : Get Sprite Palette
Sprite 9,150,50,11
M$="Anim 12,(1,4)(2,4)(3,4)(4,4)(5,4)(6,4) ; "
M$=M$+"Move 300,150,150 ; Move -300,-150,75"
Amal 9,M$
Amal On 9
Direct
```

Le deuxième exemple combine un déplacement de sprite à une animation. Remarquez que nous avons séparé les commandes par un point-virgule ";". Cette séparation permet que les deux opérations soient complètement indépendantes l'une de l'autre. Lorsque la séquence d'animation est définie, AMAL saute immédiatement à l'instruction suivante et l'animation commence.

Il est important de comprendre que Anim fonctionne seulement avec les sprites et les bobs. Donc, il n'est pas possible d'animer des écrans entiers au moyen de cette commande.

Les boucles simples

Jump *(Redirige un programme AMAL)*

J label

Cette commande permet simplement de se déplacer d'une partie d'un programme AMAL vers un autre. label est la cible de votre saut et elle doit avoir été définie dans une autre partie de votre programme courant.

Toutes les étiquettes AMAL sont définies par une seule lettre majuscule suivie d'un deux points. Comme avec les instructions, vous pouvez les faire suivre de lettres minuscules pour qu'elles soient plus lisibles. Voici quelques exemples:

```
S:
Swoop:
Label:
```

Rappelez-vous que chaque étiquette est définie par une **seule** lettre. Par conséquent, S et Swoop se rapportent en fait à la **même** étiquette! Si vous essayez de définir deux étiquettes commençant par une lettre identique, le message d'erreur *étiquette déjà définie dans la chaîne d'animation vous sera communiqué.*

Chaque programme AMAL peut être doté de son propre jeu d'étiquettes. Il est parfaitement possible d'utiliser des étiquettes semblables dans différents programmes. Exemple:

```
Load "AMOS_DATA:Sprites/octopus.abk"
Get Sprite Palette
Rem Définir sept sprites calculés en bas de l'écran
For S=8 To 20 Step 2
   Sprite S,200,(S-7)*13+40,1
Next S
Rem Créer sept programmes AMAL
For S=1 To 7
   Channel S To Sprite 6+(S*2)
   M$="Anim 0,(1,2)(2,2)(3,2)(4,2) ; Label: Move "+Str$(S*2)+",0,7 ; "
   M$=M$+"Move -" +Str$((6-2)*2)+",0,7 ; Jump Label"
   Amal S,M$
Next S
Rem OK, maintenant animez le tout!
Amal On : Direct
```

L'instruction suivante permet de déplacer un sprite vers la position courante de la souris de façon itérative:

```
Load "AMOS_DATA:Sprites/octopus.abk"
Get Sprite Palette
Sprite 8,150,150,1
Amal 8,"R: Move XM-X,YM-Y,8 ; Pause; Jump R"
Amal On 8
```

Puisque les commandes AMAL s'effectuent au moyen d'interruptions, les boucles infinies pourraient être catastrophiques. C'est pourquoi le nombre de sauts de votre programme est automatiquement compté et conservé en mémoire. Si le nombre de

sauts dépasse dix, le système AMAL ne tiendra pas compte des sauts suivants.

Remarque: Si vous vous fiez à ce système et laissez vos programmes tourner en permanence, vous allez gâcher une bonne partie de la puissance de votre Amiga. En fait, il est préférable de se limiter à un simple saut par balayage écran. Pour cela, il vous suffit d'ajouter une simple commande PAUSE avant chaque saut de votre programme. Voir PAUSE pour plus de précisions.

Variables et expressions

Let *(Affecter une valeur à un registre)*

L register=expression

L'instruction L affecte une valeur à un registre AMAL. Son action est très similaire au Basic normal, sauf que toutes les expressions sont exécutées strictement de gauche à droite.

Les registres sont des variables de nombres entiers utilisées pour représenter les valeurs intermédiaires dans vos programmes AMAL. Les nombres autorisés sont compris entre -32768 et +32767. Il existe trois types de registre de base:

Les registres internes

Chaque programme AMAL a son propre jeu de 10 registres internes. Les noms de ces registres commencent par la lettre R, suivie d'un chiffre compris entre 0 et 9 (R0-R9).

Les registres internes ressemblent aux variables locales définies dans une procédure AMOS Basic.

Les registres externes

Les registres externes sont plutôt différents parce que les valeurs qu'ils renferment sont intercalées entre des programmes AMAL distincts. Ceci vous permet de transférer des informations entre plusieurs routines AMAL. AMAL met à votre disposition 26 registres externes dont les noms sont compris entre RA et RZ.

Vous pouvez accéder directement au contenu de n'importe quel registre interne ou externe depuis votre programme Basic au moyen de la fonction AMREG (expliquée par la suite).

Les registres spéciaux

Les registres spéciaux sont un groupe de trois valeurs déterminant l'état de votre objet. *X*, Y représentent les coordonnées de votre objet. En changeant ces registres, vous pouvez promener votre objet à l'écran. Exemple:

```
Load "AMOS_DATA:Sprites/Frog_Sprites.abk" : Channel 1 To Bob 1
Flash Off : Get Sprite palette : Bob 1,0,0,1
Amal 1,"Loop: Let X=X+1 ; Let Y=Y+1; Pause; Jump Loop"
Amal On 1 : Direct
```

A mémorise le numéro de l'image qui est affichée par un sprite ou un bob. Vous pouvez

modifier cette valeur pour générer vos propres séquences d'animation comme suit:

```
Load "AMOS_DATA:Sprites/Frog_Sprites.abk" : Get Sprite Palette : Flash Off
Channel 2 to bob 1 : Bob 1,150,100,1
M$="Loop: Let A=A+1 ; "
M$=M$+"For R0=1 To 5 ; Next R0 ; Jump Loop"
Amal 2,M$
Amal On 2 : Direct
```

Nous expliquerons la boucle *For To Next* plus en détails par la suite. Cette boucle est utilisée pour ralentir chaque changement d'image du Bob 1. Lorsque la boucle *Next* est exécutée, AMAL attendra l'apparition d'un prochain balayage. Notez également l'utilisation du ";" pour séparer les instructions AMAL mais vous pouvez très bien le remplacer par un espace.

Les opérateurs

Les expressions AMAL peuvent représenter toutes les opérations arithmétiques normales, à l'exception du MOD. Vous pouvez également utiliser les opérations logiques suivantes dans vos calculs:

&	ET logique
|	OU logique

Notez qu'il n'est pas possible de changer l'ordre d'évaluation en utilisant des parenthèses "()" car elles ralentiraient considérablement vos calculs et réduiraient ainsi le temps permis dans l'interruption. Voici quelques instructions supplémentaires:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Hide
Get Sprite Palette
Sprite 8,X MOUSE,Y MOUSE,1
Amal 8,"Loop: Let X=XM ; Let Y=YM ; Pause ; Jump Loop"
Amal On 8
Direct

Load "AMOS_DATA:Sprites/octopus.abk" : Hide
Get Sprite Palette
Sprite 8,X MOUSE,Y MOUSE,1
Amal 8,"Anim 0,(1,4)(2,4)(3,4)(4,4) ; Loop: Let X=XM ; Let Y=YM ; Pause ; Jump Loop"
Amal On
Direct
```

Les exemples ci-dessus simulent en fait la commande CHANGE MOUSE. Toutefois, ce système est encore bien plus puissant puisque vous pouvez facilement déplacer les bobs, les sprites calculés ou même les écrans au moyen de cette même technique.

La prise de décisions

If *(Se brancher sur une chaîne AMAL)*

If test Jump L
Cette instruction vous permet d'effectuer de simples tests dans vos programmes AMAL. Si l'expression *test* est de -1 (vrai), le programme se branchera sur l'étiquette *L*, sinon AMAL avancera immédiatement à l'instruction suivante. Notez qu'à la différence de son équivalent en Basic, cette instruction vous permet seulement de faire un seul branchement après le test.

L'intégration de commandes en caractères minuscules comme "then" ou "else" dans cette instruction est une pratique courante. Cela clarifie l'action de la commande. Voici un exemple:

```
If X>100 then Jump Label else Let X=X+1
```

test peut être n'importe quelle expression logique que vous voulez et peut comprendre les signes suivants:

<>	signe différent
<	signe «inférieur à»
>	signe «supérieur à»
=	signe égal

Exemple:

```
Load "AMOS_DATA:Sprites/octopus.abk"
Get Sprite Palette
Sprite 8,130,50,1
C$="Main: If XM>100 Jump Test: "
C$=C$+" Let X=XM"
C$=C$+" Test: If YM>100 Jump Main "
C$=C$+" Let Y=YM Jump Main"
Amal 8,C$ : Amal On : Direct
```

Vous pouvez trouver un exemple plus important dans **4** vous permettant de contrôler la position d'un sprite au moyen du joystick. Ce moyen de contrôle est bien rudimentaire et pourrait être considérablement accélérée par la commande AUTOTEST. Voir AUTOTEST.

Attention! N'essayez pas de combiner plusieurs tests en une seule expression AMAL en utilisant &" ou "|". Puisque les expressions sont exécutées de gauche à droite, cette combinaison entraînera un message d'erreur. Prenez l'expression: X>100|Y>100. Elle a pour but de vérifier si X>100 OU Y>100. En pratique, l'expression sera exécutée dans l'ordre suivant:

X>100	Peut être VRAI ou FAUX
|Y	Combine OU et Y avec une porte OU
>100	Vérifie si (Y>100|Y)>100)

Le résultat de l'expression ci-dessus n'a évidemment pas de rapport avec la valeur escomptée. Les utilisateurs mathématiciens peuvent éviter ce problème en utilisant l'algèbre de Boole. Affectez d'abord chaque test à un seul registre AMAL comme suit:

```
Let R0=X>100; Let R1=Y>100
```

Combinez maintenant ces tests en une seule expression en utilisant "| et "&" et placez-la directement dans votre instruction If.

```
If R0 | R1 then Jump L ...
```

Cette instruction peut sembler un peu idiote mais elle marche parfaitement bien en pratique.

For To Next *(Boucle dans AMAL)*

```
For reg=start To end
: :
Next reg
```

Cette instruction met en place une boucle classique FOR...NEXT qui est presque semblable à son équivalente en Basic. Ces boucles peuvent être exploitées dans vos programmes pour déplacer des objets selon des schémas visuels complexes. reg peut être n'importe quel registre normal AMAL (R0-R9 ou RA-RZ). Cependant, vous ne pouvez pas utiliser les registres spéciaux à cette fin.

Comme avec le Basic, le nom du registre se trouvant après le Next doit correspondre à celui que vous avez spécifié dans le For, sinon vous obtiendrez un message d'erreur de syntaxe AMAL. Notez également que la taille du pas est toujours forcée à un. En outre, il est possible "d'emboîter" un nombre illimité de boucles, l'une dans l'autre.

Notez que chaque canal d'animation effectuera une seule boucle par balayage. Il permet que les effets de vos boucles et l'image soient synchrones et il vous évite d'avoir à ajouter une commande explicite Pause avant chaque Next.

Produire une série d'attaques dans un jeu

Vous pouvez vous servir de ces boucles pour créer certains schémas bien complexes de déplacement. Le type de mouvement le plus facile à effectuer est la ligne droite. Il peut être produit en utilisant une seule boucle For...Next comme suit:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,130,60,1
C$=C$+"For R0=1 To 320 ; Let X=X+1 ; Next R0 ; Jump Loop" : Rem
```

```
Déplacement d'un sprite de gauche à droite
Amal 8,C$ : Amal On 8 : Direct
```

Vous pouvez prolonger ce programme pour déplacer rapidement un objet en avant et en arrière, au travers de l'écran.

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,130,60,1
C$="Loop:For R0=1 to 320 ; Let X=X+1 ; Next R0 ; " : Rem Avancée d'un sprite
C$=C$+"For R0=1 To 320 ; Let X=X-1 ; Next R0 ; Jump Loop" : Rem Recul d'un sprite
Amal 8,C$ : Amal On 8 : Direct
```

La première boucle fait avancer l'objet de gauche à droite et le deuxième de droite à gauche.

Jusqu'à maintenant, le schéma a été limité à des déplacements horizontaux. Afin de créer une réelle série d'attaques, il est nécessaire d'intégrer un élément vertical à ce mouvement. Pour cela, il vous suffit d'intercaler une autre boucle dans votre programme.

```
Load "AMOS_DATA:Sprites/octopus.abk"
Get Sprite Palette
Sprite 8,130,60,1
C$="For R1=0 To 10 ;"
C$=C$+"For R0=1 To 320 ; Let X=X+1 ; Next R0 ;" : Rem Avancée
C$=C$+"Let Y=Y+8 ; " : Rem Descente d'un sprite à l'écran
C$=C$+"For R0=1 To 320 ; Let X=X-1 ; Next R0 ;" : Rem Recul
C$=C$+"Let Y=Y+8 ; Next R1": Rem Descente d'un sprite
Amal 8,C$:Amal On 8
Direct
```

Le programme ci-dessus produit un plan progressif d'attaques mais tout à fait simple. Vous trouverez une démonstration supplémentaire dans le fichier **Exemple 14.1** du dossier MANUEL.

Enregistrer une séquence de déplacements complexes

PLay

PLay path

Si vous avez déjà regardé les séries progressives d'attaques d'un jeu d'arcade moderne, vous avez peut-être pensé que leur manipulation ne sera jamais dans vos moyens mais réfléchissez. La commande AMAL PLay vous permet d'animer librement vos objets dans pratiquement n'importe quelle séquence de déplacements que vous

pouvez imaginer. Elle s'exécute selon un schéma de déplacement préalablement défini et conservé dans la banque de mémoire d'AMAL.

Ces schémas sont créés par l'accessoire AMAL de la disquette de programme AMOS. Cet accessoire mémorise simplement une séquence de déplacements de la souris et les fait entrer directement dans la banque de mémoire d'AMAL. Dès que vous avez défini vos schémas de cette façon, vous pouvez facilement les affecter à n'importe quel objet à l'écran, en reproduisant parfaitement vos schémas d'origine. Vous pouvez modifier la vitesse et la direction de votre mouvement à tout moment depuis votre programme AMOS Basic.

Lorsque AMAL rencontre une commande PLay pour la première fois, il recherche le déplacement enregistré que vous avez spécifié dans la banque AMAL au moyen du paramètre *path*. *path* est simplement un nombre compris entre un et le nombre maximal de schémas contenus dans la banque. Si un problème surgit pendant cette phase, AMAL suspendra l'instruction PLay et passera à la prochaine instruction de votre chaîne d'animation.

Une fois le schéma initialisé, le registre R0 est chargé du tempo de déplacement. Celui-ci détermine l'intervalle de temps entre chaque pas de déplacement. Les durées sont toutes mesurées en unités de 50ème par seconde. En changeant ce registre dans votre programme AMAL, vous pouvez accélérez ou ralentir les déplacements de vos objets.

Notez que chaque pas de déplacement est **ajouté** aux coordonnées courantes de votre objet. Donc, si un objet est déplacé par la suite au moyen des instructions Sprite ou Bob, il continuera ses manoeuvres en partant de la nouvelle position de l'écran, sans être affecté. Il est donc possible d'animer des douzaines d'objets différents à l'écran au moyen d'une seule séquence de déplacements.

Le registre R1 contient un code déterminant la direction de vos déplacements. Il existe trois versions possibles:

• R1>0 En avant

Une valeur de un attribuée à R1 spécifie que le schéma de déplacement sera réexécuté du début à la fin, exactement dans son ordre d'origine. (Défaut)

• R1=0 En arrière

Dans de nombreuses séquences d'animation, vous pouvez avoir besoin de faire reculer et avancer vos objets au travers de l'écran selon un schéma complexe. Pour leur faire changer de direction, il vous suffit d'affecter R1 d'un zéro. Votre objet fera un demi-tour et exécutera vos déplacements d'origine à l'envers.

• R1=-1 Sortie

Si une collision est détectée dans votre programme AMOS, il vous faudra arrêter votre objet et produire un effet d'explosion. Pour cela, forcez R1 à une valeur de moins un. AMAL suspendra alors l'instruction de jeu et se branchera immédiatement à l'instruction suivante de votre séquence d'animation.

Ces registres sont ingénieux parce que vous pouvez les changer directement

depuis AMOS Basic. Ils vous permettent ainsi de contrôler les schémas de déplacement directement depuis votre programme principal. Il existe même une instruction spéciale AMPLAY pour vous faciliter les choses.

La commande PLay est parfaite pour contrôler les ennemis dans un jeu d'arcade. En fait, c'est l'instruction la plus puissante d'AMAL.

AMAL *(Appeler un programme AMAL)*

AMAL n,a$
AMAL n,p
AMAL n,a$ to address

La commande AMAL affecte un programme AMAL à un canal d'animation. Ce programme peut être saisi depuis une chaîne en a$ ou directement depuis la banque AMAL.

La première version de l'instruction charge votre programme depuis la chaîne *a$* et l'affecte au canal *n*. *a$* peut contenir n'importe quelle liste d'instructions AMAL. Vous pouvez également charger votre programme d'une banque de mémoire créée en utilisant l'accessoire AMAL. *p* se rapporte maintenant au numéro du programme AMAL enregistré dans la banque numéro 4.

n est le numéro du canal d'animation compris entre 0 et 63. Chaque canal AMOS peut être affecté de façon indépendante à un bob, à un sprite ou à un écran.

Seuls, les 16 premiers programmes AMAL peuvent être effectués sans interruptions. Pour les suivants, il vous faudra exécuter vos programmes directement depuis le Basic en utilisant la commande SYNCHRO.

La version finale de l'instruction AMAL est destinée aux utilisateurs avertis. Au lieu de déplacer un objet réel, elle copie simplement le contenu des registres X,Y et A dans une zone spécifique de la mémoire. Vous pouvez alors utiliser cette information directement dans vos propres routines Basic. Il est donc possible d'exploiter le système AMAL pour animer tout ce que vous voulez, depuis un bloc jusqu'à un simple caractère. La syntaxe est la suivante:

AMAL n, a$ To adress

address doit être un nombre PAIR et doit désigner une zone sûre de la mémoire, de préférence dans une chaîne AMOS ou une banque de mémoire. A chaque exécution du programme AMAL (50 fois par seconde), les valeurs suivantes seront inscrites dans la zone mémoire:

Adresse	**Effet**
Adresse	Bit 0 est forcé à 1 si le X a changé. Bit 1 indique que Y a été modifié. Bit 2 sera forcé à 1 si l'image (A) a changé depuis la dernière interruption.
Adresse+2	Est un **mot** contenant la dernière valeur de X
Adresse+4	Contient la valeur courante de Y
Adresse+6	Stocke la valeur de A.

Vous pouvez accéder à ces valeurs depuis votre programme en utilisant une simple instruction DEEK.
Attention: Cette option écrase complètement toutes les affectations précédentes permises par CHANNEL.

Les commandes AMAL

Voici une liste complète des commandes AMAL proposées:

Move

Cette commande déplace progressivement un objet d'un point à un autre. Sa syntaxe est:

Move deltaX, deltaY, steps

deltaX représente la distance horizontale de déplacement. Les nombres positifs indiquent un déplacement de gauche à droite et les valeurs négatives, un déplacement de droite à gauche.

deltaY spécifie la déplacement vertical. Si *deltaY* est positif, alors votre objet descendra à l'écran, sinon il sera poussé vers le haut.

n indique le nombre de pas nécessaires pour effectuer le déplacement. Les déplacements les plus progressifs sont générés lorsque *deltaX* et *deltaY* sont d'exacts multiples de n.

A (Anim)

Anim cycles,(image,durée)(image,durée)...

L'instruction Anim affecte une séquence d'images à un sprite ou à un objet de manipulation graphique pour produire un effet réel d'animation.

cycles spécifie le nombre de répétitions de l'animation. S'il est forcé à zéro, l'animation va continuer indéfiniment. image affecte un numéro à chaque *image* de votre animation. *durée* définit la durée (en 50ème de seconde) d'affichage de l'image.

Après que la commande Anim ait été initialisée, AMAL se branchera automatiquement sur l'instruction suivante. Elle vous permet de combiner à la fois l'animation et le déplacement dans le même programme AMAL.

Let

Let reg=exp

Cette commande affecte une valeur à un registre AMAL. *reg* est le nom du registre AMAL qui doit être changé. Il existe 10 registres internes compris entre R0 et R9 et 26 autres registres externes (de RA à RZ). Vous pouvez également modifier la position et inscrire le type de votre objet directement en utilisant les registres spéciaux X, Y et A.

expr est une expression arithmétique courante et elle est exécutée de gauche à droite pour produire le résultat final.

La plupart des opérateurs courants sont supportés, y compris _,-,* et /. Cependant, vous n'êtes pas autorisé à changer l'ordre de calcul en utilisant des parenthèses "()".

Jump

Jump L

La commande Jump saute du point courant de votre programme AMAL à l'étiquette *L*. L est le nom de l'étiquette préalablement définie dans votre chaîne AMAL. Les étiquettes sont composées d'une seule lettre majuscule et sont créées en utilisant un "." comme dans le Basic courant.

If

If exp Jump L

L'instruction If vous permet de sauter d'une partie d'un programme AMAL à une autre en fonction du résultat d'un test. *exp* est une expression logique exprimée dans la syntaxe courante.

Si *exp* est VRAI, alors le programme se branchera sur l'étiquette *L*, sinon elle exécutera immédiatement l'instruction suivante après le Jump.

Il existe deux autres variantes de cette commande utilisées par la fonction AUTOTEST:

```
If exp Direct L   (Choisit une partie du programme devant être exécuté après un autotest)
If exp eXit       (Sort d'Autotest)
```

Voir AUTOTEST pour de plus amples précisions.

For To Next

For Reg=start To end ...Next Reg

C'est une application directe des boucles Basic FOR...NEXT. Reg peut être n'importe quel registre AMAL interne ou externe. Comme d'habitude, les boucles peuvent être emboîtées mais le pas de progression de votre boucle est toujours forcée à un.

Notez qu'AMAL attendra automatiquement le prochain balayage avant de revenir au début de votre boucle avec Next. Puisque les déplacements d'objets de votre programme seront seulement visualisés à l'actualisation de l'écran, après l'impulsion de retour de balayage, les boucles plus rapides gâcheraient simplement un temps de traitement précieux sans obtenir d'effets visibles. Ainsi, vos boucles For...Next sont automatiquement synchrones avec les actualisations de l'écran pour produire des effets les plus progressifs possibles.

PLay

PLay path

La commande PL anime vos objets au moyen d'une série de déplacements mémorisée dans la banque AMAL. Vous pouvez entrer les schémas directement à l'aide de la souris, en utilisant l'utilitaire AMAL- Editor. Aucune limite n'est imposée au type de déplacements que vous pouvez produire avec ce système.

path est le numéro d'un schéma préalablement sauvegardé dans la banque AMAL. Si ce schéma n'existe pas, AMAL sautera l'instruction PL et se branchera immédiatement à la commande suivante de votre séquence d'animation.

Tous les déplacements sont effectués en fonction de la position courante de vos objets. Il est donc possible de déplacer toute une série d'attaques en utilisant une seule définition de chemin. Vous pouvez également déplacer un objet directement en Basic sans affecter du moins le déplacement. L'état du déplacement courant est contrôlés au moyen des deux registres AMAL.

R0 contient le tempo de votre déplacement. L'augmentation de cette valeur accélèrera l'objet à l'écran.

R1 contient la direction du déplacement. Il existe trois autres alternatives.

R1>0	Exécute la séquence de déplacement dans l'ordre d'origine.
R1=0	Exécute les pas de votre déplacement dans l'ordre inverse.
R1=-1	Arrête la séquence de déplacement et passe à l'instruction AMAL suivante.

Vous pouvez changer le contenu de ces registres à tout moment dans votre programme Basic en utilisant la commande AMREG ou la commande spéciale AMPLAY.

Une explication plus détaillée vous est donnée dans le guide d'initiation AMAL vers le début de ce chapitre. Référez-vous également **au fichier Exemple 14.2** du dossier MANUEL.

Attention: Il est important d'utiliser le point virgule pour séparer vos instructions AMAL. La chaîne suivante produira un message d'erreur *Banque AMAL non réservée* simplement en raison de l'inexistence d'un caractère de séparation.

```
A$="Pause Let R0=1"
```

La syntaxe correcte est:

```
A$="Pause ; Let R0=1"
```

End

End

Cette commmande met fin au programme AMAL tout entier et désactive la fonction Autotest si elle a été définie.

Pause

Pause
Cette commande suspend temporairement l'exécution de votre programme AMAL et attend la prochaine impulsion de balayage. Votre programme sera ensuite automatiquement repris à partir de l'instruction suivante.

Pause est souvent utilisée avant une commande Jump pour s'assurer que le nombre de branchements par impulsion de balayage est inférieur à 10. Elle permet de libèrer le temps précieux de traitement de vos programmes Basic et peut influencer considérablement leur vitesse dans l'ensemble. Il est donc plus efficace de faire précéder vos commandes Jump d'une instruction Pause.

AUtotest

AU (List of tests)

La fonction Autotest d'AMAL a été conçue pour permettre une interaction rapide entre AMAL et l'utilisateur. Elle ajoute un test spécial au début du programme AMAL qui est effectué à chaque balayage avant que le reste du programme ne soit exécuté. Référez-vous au système Autotest pour plus de précisions.

eXist

eXit

Cette commande vous fait sortir d'Autotest et relance le programme AMAL courant.

Wait

Wait

Cette commande fige votre programme AMAL et exécute seulement l'Autotest.

On

On

ON active le programme principal après une commande Wait.

Direct

Direct

Cette commande définit la section du programme principal qui doit être exécutée après un autotest.

Fonctions AMAL

=XM *(Renvoyer l'abscisse de la souris)*

Cette fonction est exactement identique à la fonction X MOUSE en AMOS Basic. Elle renvoie l'abscisse du curseur de la souris en coordonnées machine.

=YM *(Renvoyer l'ordonnée de la souris)*

Cette fonction renvoie l'ordonnée de la flèche de la souris en coordonnées machine.

=K1 *(Etat du bouton gauche de la souris)*

K1 renvoie une valeur de -1 (vrai) si le bouton gauche de la souris a été abaissé, sinon la valeur 0 (faux) est communiquée.

=K2 *(Etat du bouton droit de la souris)*

K2 renvoie l'état du bouton droit de la souris. Si le bouton a été abaissé, alors K2 renverra la valeur -1 (vrai).

=J0 (Tester le joystick droit)

La fonction J0 teste le joystick droit et renvoie une configuration binaire contenant l'état courant. Voir JOY pour plus de précisions.

=J1 *(Tester le joystick gauche)*

La fonction J1 teste le joystick gauche et renvoie une configuration binaire dans la syntaxe courante.

=Z(n) *(Nombre aléatoire)*

La fonction Z renvoie un nombre aléatoire compris entre -32767 et 32768. Ce nombre peut être limité à une plage spécifique au moyen du masque binaire *n*.

Une opération ET logique est effectuée entre le masque binaire n et le nombre aléatoire pour donner le résultat final. Par exemple, si n est forcé à une valeur de 255, les nombres renvoyés seront compris entre 0 et 255.

Puisque cette fonction a été optimisée pour être plus rapide, le nombre renvoyé n'est pas complètement aléatoire. S'il vous faut vraiment des nombres aléatoires, il serait préférable de produire vos valeurs en utilisant le RND du Basic et en les chargeant dans un registre externe AMAL au moyen de la fonction AMREG.

=XH *(Convertir une abscisse écran en une abscisse machine)*

=XH(s,x)
Cette commande convertit une abscisse écran en son abscisse machine équivalente se rapportant à l'écran *s*.

=YH *(Convertir une ordonnée écran en une ordonnée machine)*

=YH(s,y)

Cette commande transforme une ordonnée écran en une ordonnée machine se rapportant à l'écran *s*.

=XS *(Conversion des abscisses machine en abscisses écran)*

=XS(s,x)

Cette commande change l'abscisse machine en une abscisse graphique se rapportant à l'écran *s*.

=YS *(Conversion des abscisses machine en abscisses écran)*

Cette commande transforme l'ordonnée machine en son ordonnée écran équivalente.

=BC *(Vérifier les collisions entre bobs)*

=Bob Col(n,s,e)

Cette commande est identique à l'instruction équivalente BOB COL de l'AMOS Basic. Elle recherche les collisions entre le bob numéro *n* et les bobs de s à e.

Si une collision est détectée, alors BC renvoie la valeur -1 (vrai), sinon la valeur 0 est renvoyée (faux). Il se peut que cette instruction **ne** soit **pas** exécutée pendant une interruption. Par conséquent, elle est seulement possible si vous exécutez vos routines AMAL directement depuis le Basic au moyen de l'instruction SYNCHRO.

SC(m,s,e) *(Collisions de sprites)*

=Sprite Col(n,s,e)

Cette commande est équivalente à la fonction SPRITE COL en AMOS Basic. Elle recherche les collisions entre le sprite *n* et les sprites de *s* et *e*. Si le test s'avère positif, la valeur -1 (vrai) sera renvoyée. Comme avec la fonction précédente BC, elle est

seulement permise avec l'instruction SYNCHRO.

=C(n) *(Col)*

Cette commande renvoie l'état de l'objet n après validation de la fonction SC ou BC. S'il y a collision, cette fonction renvoie la valeur -1 (vrai), sinon la valeur zéro est communiquée (faux).

=V(v) *(Vumètre)*

La fonction VU examine un des canaux musicaux et renvoie l'intensité du son courant. C'est un nombre compris entre 0 et 63. Vous pouvez utiliser cette information pour animer vos objets en synchronisme avec la musique. Vous trouverez une illustration de cette fonction dans le fichier **Exemple 14.3.** Référez-vous également à la fonction VUMETER d'AMOS Basic.

Contrôler l'AMAL depuis le Basic

AMAL ON/OFF *(Commencer/arrêter un programme AMAL)*

AMAL ON[n]

Après avoir défini votre programme AMAL, il vous faut l'exécuter au moyen de la commande AMAL ON. Cette dernière active le système AMAL et commence vos programmes à partir de la première instruction.

AMAL ON active tous vos programmes. Le paramètre optionnel *n* vous permet de commencer une seule routine à la fois.

AMAL OFF[n]

Cette commande arrête l'exécution d'un ou de tous les programmes AMAL. Ces programmes sont effacés de la mémoire. Vous pouvez seulement les relancer en les redéfinissant au moyen de l'instruction AMAL.

AMAL FREEZE *(Figer temporairement un programme AMAL)*

AMAL Freeze [n]

Cette commande arrête le déroulement d'un ou plusieurs programmes AMAL. Vous pouvez relancer vos programmes à votre convenance au moyen d'un simple appel à AMAL ON. Notez que cette instruction doit toujours être utilisée pour arrêter AMAL avant d'exécuter une commande telle que DIR, sinon des problèmes de synchronisation peuvent entraîner un fouilli à l'écran.

=AMREG= *(Obtenir la valeur d'un registre AMAL externe)*

r=AMREG([canal],n)
AMREG([canal],n)=expression

La fonction AMREG vous permet d'accéder au contenu des registres AMAL interne et externe directement depuis le corps de votre programme Basic.

n est le numéro du registre. Les valeurs possibles sont comprises entre 0 et 25, zéro représentant le registre RA et 25 indiquant le registre RZ.

En utilisant le paramètre optionnel *canal*, vous pouvez désigner n'importe quel registre AMAL interne. Dans ce mode, *n* est compris entre 0 et 9, représentant les registres R0 à R9.

L'exemple suivant montre comment il est possible de retrouver la position X courante d'un sprite depuis le Basic:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Channel 1 To Sprite 8 : Sprite 8,100,100,1
A$="Loop: Let RX=X+1; Let X=RX; Pause; Jump Loop" : Rem X deviendra trop grand si >640
Amal 1,A$ : Amal On : Curs Off
Do
   Locate 0,0
   Z=Asc("X")-65 : Rem Notez l'utilisation de ASC pour obtenir le numéro de registre
   Print Amreg(Asc("X")-65)
Loop
```

AMPLAY *(Contrôler une animation produite au moyen de PLay)*

AMPLAY tempo, direction[start TO end)]

Toutes les séquences que vous avez produites au moyen de la commande AMAL PL sont contrôlées par les registres internes R0 et R1. Chaque objet sera affecté de son propre jeu de registres AMAL. Par conséquent, si vous animez plusieurs objets, il vous faudra souvent initialiser plus d'un registre aux mêmes valeurs.

Bien que cette initialisation puisse être accomplie au moyen de la classique fonction AMREG, il serait évidemment bien plus facile d'avoir une seule instruction permettant de remplacer à la fois R0 et R1 par toute une série d'objets. C'est le but de la commande AMPLAY.

AMPLAY saisit le *tempo* et la *direction* de vos déplacements et les charge dans les registres R0 et R1 contrôlant les canaux sélectionnés.

tempo contrôle la vitesse de votre objet à l'écran et définit une temporisation (en 50ème de seconde) entre chaque pas de déplacement successif.

direction change la direction du déplacement. Voici une liste des diverses options.

Valeur	Direction du déplacement
>0	Déplace l'objet sélectionné dans la direction initiale.
0	Inverse le déplacement et fait reculer l'objet
-1	Suspend le schéma de déplacement et se branche à l'instruction suivante de votre séquence d'animation AMAL.

Par défaut, cette instruction affecte tous les canaux courants d'animation. Vous pouvez modifier cette affectation en ajoutant quelques points explicites *premier* et *dernier* à la commande. *premier* est le numéro du canal à affecter au premier objet. *dernier* représente le numéro du canal affecté au dernier objet de votre liste.

Notez que vous pouvez omettre le *tempo* ou la *direction* si besoin. Exemples:

```
Amplay ,0 : Rem Inverser vos objets
Amplay 2, : Rem Ralentir les schémas de déplacement
Amplay 3,1 : Rem Mettre le tempo sur trois et la direction sur 1
Amplay ,-1 3 To 6 : Rem Arrêter les déplacements sur les canaux 3,4,5 et 6
```

=CHANAN *(Tester une animation Amal)*

s=CHANAN(canal)

Cette fonction simple vérifie l'état d'une séquence d'animation AMAL et renvoie -1 (vrai) si celui-ci est couramment actif ou 0 (faux) si l'animation est terminée. *canal* représente le numéro du canal à tester. Voici un exemple:

```
Load "AMOS_DATA:Sprites/Monkey_right.abk" : Get Sprite Palette
Sprite 9,150,150,11
M$="Anim 12, (11,4)(12,4)(13,4)(14,4)(15,4)(16,4);"
Amal 9,M$ : Amal On
While Chanan(9)
Wend
Print "Animation terminée"
```

=CHANMV *(Vérifier si un objet continue à se déplacer)*

s=CHANMV(channel)

Cette commande renvoie valeur -1 (vrai) si l'objet affecté à *canal* est en train de se déplacer, et la valeur 0 (faux) sinon.
Vous pouvez utiliser cette commande avec l'instruction AMAL Move pour vérifier si une séquence de commandes a terminé l'exécution de tous ses pas. Vous pouvez alors relancer la séquence à la nouvelle position à l'aide d'une chaîne de déplacement appropriée si besoin. Exemple:

```
Load "AMOS_DATA:Sprites/Monkey_right.abk" : Get Sprite Palette
Sprite 9,150,150,11
M$="Move 300, 150, 150; Move -300,-150,75"
Amal 9,M$ : Amal On
While Chanan(9)
Wend
Print "Déplacement terminé"
```

Les erreurs AMAL

=AMALERR *(Renvoyer la position d'une erreur)*

p=AMALERR

Cette commande renvoie la position de la chaîne d'animation courante si une erreur se produit. Un examen minutieux de cette chaîne vous permettra de corriger rapidement vos erreurs. Exemple:

```
Load "AMOS_DATA:Sprites/octopus.abk"
Sprite 8,100,100,1
A$="L: IF X=300 then Jump L else X=X+1; Jump L"
Amal 8,A$
```

Ce programme produira une erreur de syntaxe parce que IF est interprété comme étant deux instructions "I" et "F". Pour trouver la position de l'erreur dans la chaîne d'animation, entrez l'instruction suivante depuis la fenêtre directe:

```
Print Mid$(A$,Amalerr,Amalerr+5)
```

Messages d'erreur

Si vous commettez une erreur dans un de vos programmes AMAL, AMOS sortira du Basic en vous communiquant le message d'erreur approprié. Voici une liste complète des erreurs pouvant être produites par ce système et une explication de leur cause la plus probable.

Banque non réservée: Cette erreur est causée par l'appel de l'instruction PLay sans avoir prélablement chargé une banque contenant les données du déplacement en mémoire. Cette banque doit être créée à l'aide du programme accessoire AMAL. Si vous n'utilisez jamais PLay, vérifiez alors que vous avez séparé correctement les instructions Pause et **L**et dans votre programme.

Instruction seulement valide dans autotest: Vous avez appelé par inadvertance les instructions Direct ou eXit depuis le programme principal AMAL.

Instruction interdite dans Autotest: Autotest peut être seulement utilisée avec un nombre limité de commandes. Il n'est pas possible de déplacer ou d'animer vos objets de n'importe quelle façon dans autotest. Recherchez donc les fausses commandes

comme Move, Anim ou For...Next.

Se brancher vers/à l'intérieur d'Autotest dans une chaîne d'animation: Les commandes se trouvant dans une fonction autotest sont complètement indépendantes de votre principal programme AMAL. Par conséquent, AMAL ne vous permet pas de vous brancher directement dans une procédure AUtotest. Pour sortir d'un autotest et revenir à votre programme principal AMAL, vous devez utiliser eXit ou Direct.

Etiquette déjà définie dans une chaîne d'animation: Vous avez essayé de définir deux fois la même étiquette dans votre programme AMAL. Toutes les étiquettes AMAL comportent une seule lettre MAJUSCULE. Par conséquent, **Test** et **Total** sont simplement deux versions différentes de la même étiquette (**T**). Cette erreur se produit également si vous séparez, par mégarde, deux instructions par un ":" (deux points). Utilisez plutôt un point-virgule.

Etiquette non définie dans une chaîne d'animation: Cette erreur se produit lorsque vous essayez de vous brancher sur une étiquette qui n'existe en fait pas dans votre chaîne d'animation.

Next sans For dans une chaîne d'animation: Comme avec son équivalent en Basic, chaque commande For doit être accompagnée de son instruction correspondante Next. Recherchez les fausses commandes Next dans les boucles emboîtées.

Erreur de syntaxe dans une chaîne d'animation: Vous avez fait une erreur de frappe dans une de vos chaînes d'animation. Il est facile de commettre cette erreur par mégarde en entrant une instruction AMAL en entier, comme avec son équivalent en Basic. Rappelez-vous que les commandes AMAL comportent seulement une ou plusieurs lettres MAJUSCULES. Par conséquent, si vous essayez d'entrer des instructions telles que FOR ou NEXT, vous obtiendrez un message d'erreur. La syntaxe correcte de ces commandes sont For...Next.

Les canaux d'animation

AMOS vous permet d'exécuter simultanément 64 différents programmes AMAL. Chaque programme est affecté d'un *canal* d'animation spécifique.

Seuls, l'animation des 16 premiers canaux peut s'effectuer en utilisant des interruptions. Si vous devez animer davantage d'objets, il vous faudra désactiver les interruptions en utilisant la commande SYNCHRO OFF. Vous pouvez alors exécuter les programmes AMAL pas-à-pas en faisant un appel explicite à la commande SYNCHRO dans la boucle de votre programme principal. Par défaut, tous les canaux d'interruption sont affectés au sprite machine approprié.

CHANNEL *(Affecter un objet à un canal AMAL)*

CHANNEL n TO object s

La commande CHANNEL affecte un canal d'animation à un *objet* se rapportant à un

écran particulier. En AMAL, vous n'êtes pas limité à un seul canal par objet. Vous pouvez animer sans risque n'importe quel objet à l'écran en utilisant plusieurs canaux si besoin. Il existe plusieurs variantes de cette instruction.

Animer un sprite calculé

CHANNEL n TO SPRITE s

Cette commande affecte un sprite numéro s au canal *n*. Par défaut, les canaux de 0 à 7 sont affectés au sprite machine équivalent et ceux de 8 à 15 sont réservés aux sprites calculés appropriés.

Pour animer les sprites calculés à partir du numéro 16, il vous faut les affecter directement à un canal d'animation en utilisant la commande CHANNEL. Comme d'habitude, les numéros de sprite de 8 à 63 spécifient un sprite calculé et non un seul sprite machine. Par exemple:

```
Channel 5 To Sprite 8:Rem Anime le sprite calculé 8 en utilisant le canal 5.
```

Les registres X,Y de votre programme AMAL se rapportent maintenant aux coordonnées machine du sprite sélectionné. Similairement, l'image courante du sprite est contenue dans le registre A.

Animer un bob

Vous pouvez également utiliser les programmes AMAL pour animer les bobs.

CHANNEL n TO BOB b

Cette commande affecte un bob b au canal d'animation *n*. Ce bob sera considéré de la même façon que le sprite machine équivalent. La seule différence est que les registres X et Y contiennent maintenant la position de votre bob en coordonnées **écran.**

Notez que si vous avez activé la fonction bascule d'écran au moyen de la commande DOUBLE BUFFER, cette fonction sera automatiquement utilisée pour toutes les animations de bob. Reportez-vous à **8** du dossier MANUEL pour obtenir une illustration complète de cette commande.

Déplacer un écran

AMOS Basic vous permet de positionner facilement l'écran courant sur votre écran de télé. Normalement, vous sélectionnez l'instruction SCREEN DISPLAY. Cependant, il est quelques fois utile de pouvoir déplacer l'écran en utilisant des interruptions.

CHANNEL n TO SCREEN DISPLAY d

Cette commande positionne le canal *n* à l'écran numéro *d*. L'écran d peut être défini où vous voulez dans votre programme. Vous obtiendrez seulement un message d'erreur si l'écran n'a pas été ouvert lorsque vous lancez votre animation.

Les variables X et Y d'AMAL représentent maintenant la position de votre écran en coordonnées machine. Le registre A n'est **pas** utilisé par cette option et vous ne pouvez pas animer vos écrans au moyen d'Anim. Vous pouvez autrement effectuer toutes les instructions AMAL normales comme d'habitude. Par conséquent, vous pouvez facilement utiliser ce système pour faire "rebondir" l'image à l'écran. Exemples:

```
Load Iff "AMOS_DATA : IFF/Amospic.IFF",1
Channel 0 To screen display 1
Amal 0,"Loop: Move 0,200,100 ; Move 0,-200,100 ; Jump Loop"
Amal On : Direct

Load Iff "AMOS_DATA : IFF/Frog_Screen.IFF",1
Channel 0 to screen display 1
Rem L'écran peut seulement être affiché à certaines positions en X
Amal 0,"Loop: Let X=XM; Let Y=YM; Pause; Jump Loop"
Amal On : Direct
```

Si vous souhaitez étudier une autre illustration de cette technique, chargez le fichier **Exemple 14.4** du dossier MANUEL. Cet exemple vous montre comment utiliser SCREEN DISPLAY avec les commandes du menu pour faire monter ou descendre l'écran du menu. Elle est similaire au système d'affichage utilisé par l'excellente série d'aventure de Magnetic Scroll.

Le scrolling machine

Bien que le scrolling hardware puisse être effectué au moyen de la commande SCREEN OFFSET d'AMOS Basic, il est souvent plus facile d'animer les écrans au moyen de l'AMAL puisque ce dernier produit un effet bien plus progressif.

CHANNEL n TO SCREEN OFFSET d

Cette instruction affecte le programme AMAL numéro *n* à un écran *d* pour la réalisation du hardware scrolling. Les registres X et Y se rapportent maintenant à la portion de l'écran devant être affiché à votre télé. Le changement de ces registres fera défiler la zone visible de l'écran.
Voici un exemple:

```
Screen Open 0,320,500,32,lowres : Rem Ouvrir un autre grand écran
Screen Display 0,,45,320,250
Load Iff "AMOS_DATA:IFF/Magic_Screen.IFF"
Screen Copy 0,0,0,320,250 To 0,0,251
Screen 0 : Flash off : Get palette (0)
Channel 0 to Screen Offset 0
Amal 0,"Loop: Let X=XM-128; Let Y=YM-45; Pause; Jump Loop"
Amal On : Wait Key
```

Ce programme vous permet de faire défiler l'écran au moyen de la souris. Essayez de

déplacer la souris en mode direct. Vous trouverez une autre illustration du hardware scrolling dans le fichier **Exemple 14.5.**

Changer la taille de l'écran

CHANNEL n TO SCREEN SIZE s.

Cette instruction vous permet de changer la taille d'un écran en utilisant l'AMAL. s est le numéro de l'écran à manipuler. Les registres X et Y contrôlent respectivement la largeur et la hauteur de votre écran. Ils sont similaires aux paramètres L et H utilisés par la commande SCREEN DISPLAY. Exemple:

```
Load Iff "AMOS_DATA : IFF/Amospic.IFF",0
Channel 0 To Screen Size 0
Screen Display 0,,,320,1 : Rem Forcer la taille de l'écran à 1
A$="Loop: For R0=0 To 255 ; Let Y=R0 ; Next R0; "
A$=A$+"For R0=0 To 254; Let Y=255-R0; Next R0; J Loop"
Amal 0,A$ : Amal On : Direct
```

Arcs-en-ciel

CHANNEL n TO RAINBOW r

Cette option produit un arc-en-ciel dans un programme AMAL. Comme d'habitude, *n* est le numéro du canal d'animation compris entre 0 et 63. *r* est le numéro d'identification de votre arc-en-ciel (0-3).

X représente la BASE courante de votre arc-en-ciel. Elle représente la première couleur de la palette de votre arc-en-ciel qui sera affichée. Si vous changez cette couleur, l'arc-en-ciel paraîtra tourner. Y représente la ligne à l'écran depuis laquelle l'arc-en-ciel commence. Si vous modifiez cette valeur, l'arc-en-ciel montera ou descendra. Toutes les coordonnées sont exprimées en coordonnées **machine**.

Le registre A enregistre la hauteur de votre arc-en-ciel à l'écran. Vous trouverez une démonstration de ce système dans **11.** Voir la commande RAINBOW d'AMOS Basic pour plus de précisions.

Les Techniques avancées

Le système AUTOTEST

Normalement, tous les programmes AMAL sont effectués dans un ordre bien déterminé du début à la fin. L'exécution de certaines commandes telles que Move et For...Next prendra bien évidemment plusieurs secondes. Bien que cela n'ait pas d'importance dans la grande majorité des cas, d'importants retards peuvent être ainsi causés lors du déroulement de certains programmes. Prenez le simple programme suivant:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
```

```
Sprite 8,130,50,1
Amal 8,"Loop: Let R0=XM-X; Let R1=YM-Y; Move R0,R1,50; Jump Loop"
Amal On : Direct
```

Au fur et à mesure que vous déplacez la souris, le sprite doit la suivre à l'écran. Toutefois, en pratique, le temps de réponse est bien lent parce que les nouvelles valeurs de XM et YM sont seulement entrées après que le sprite ait terminé son déplacement. Essayez de déplacer la souris en rond. L'octopus est complètement déboussolé!

Autotest solutionne le problème en effectuant vos tests au début de chaque impulsion de retour de balayage, avant de continuer le programme courant. Vos tests se produisent alors à des intervalles réguliers de 1/50, vous donnant une réponse pratiquement instantanée!

Les commandes Autotest

La syntaxe d'Autotest est:

AUtotest (tests)

tests peut comprend n'importe quelle des commandes AMAL suivantes.

Let

L reg=exp

C'est l'instruction normale Let d'AMAL. Elle affecte le résultat d'une expression au registre *reg*.

Jump

Jump label

La commande Jump se branche sur une autre partie de l'autotest courant. Label est définie au moyen du deux points ":" et **doit** se trouver à l'intérieur des parenthèses d'autotest.

e**X**it

Cette commande vous fait sortir de l'autotest et relance le programme principal au point où il a été suspendu.

Wait

Cette commande désactive le programme principal AMAL et exécute seulement

l'Autotest.

If

Afin de simplifier le procédé de test à l'intérieur d'une routine autotest, il existe une version spécialement prolongée de l'instruction AMAL If. Elle vous permet d'effectuer une des trois actions en fonction du résultat de l'expression logique *exp*.

```
If exp Jump L     (Se branche à une autre partie de l'autotest)
If exp Direct L   (Choisit une partie du programme à exécuter après un autotest)
If exp eXit       (Sort d'autotest)
```

On

Cette commande relance le programme principal après une instruction Wait. Cela vous permet d'attendre un évènement spécifique tel que le déclic de la souris sans gâcher un temps de traitement précieux.

Direct

Direct label

Cette commande change le point à partir duquel le programme sera repris après votre test. AMAL se branchera alors automatiquement à ce point à la prochaine impulsion de suppression image. Notez que *label* **doit** être défini en dehors des parenthèses de l'Autotest.

A l'intérieur d'Autotest

Voici l'exemple précédent réécrit en utilisant la fonction Autotest.

```
Load "AMOS_DATA:Sprites/octopus.abk"
Sprite 8,130,50,1 : Get Sprite Palette
A$="AUtotest (If R0<>XM Jump Update"
A$=A$+"If R1<>YM Jump Update else eXit"
A$=A$+"Update: Let R0=XM; Let R1=YM; Direct M)" : Rem Fin d'autotest
A$=A$+"M: Move R0-X,R1-Y,20 Wait;" : Rem Essayez de remplacer 20 par d'autres valeurs!
Amal 8,A$: Amal on
```

Le sprite suit progressivement la souris, peu importe la vitesse de déplacement de cette dernière. L'action de ce programme est la suivante:

Les coordonnées de la souris sont testée chaque 50ème de seconde au moyen des fonctions XM et YM. Si elles sont inchangées depuis le dernier test, l'exécution d'Autotest est suspendue par la commande eXit. Le programme principal reprend précisément au point où il a été suspendu.

Toutefois, si la souris a été déplacée, la routine de l'autotest relance le programme principal depuis le début (étiquette **M**) en utilisant les nouvelles coordonnées

respectivement en XM et YM.

Etude de la synchronisation

UPDATE EVERY *(Economiser du temps dans vos programmes Basic)*

UPDATE EVERY n

Bien que les programmes AMAL soient pratiquement exécutés instantanément, tous les objets qu'ils manipulent doivent être explicitement tracés à l'écran de l'Amiga.

Le temps nécessaire à l'exécution de cette procédure d'actualisation n'est pas mesurable et peut varier au cours de votre programme. Cela peut entraîner une instabilité ennuyeuse dans le schéma de déplacement de certains objets.

La commande UPDATE EVERY ralentit le processus d'actualisation afin que même les plus grands objets puissent être retracés lors d'une seule actualisation de l'écran. Cela régule le système d'animation et produit des effets de déplacement très progressifs.

n est le nombre d'impulsions de retour de balayage (50ème de seconde) entre chaque actualisation d'écran. En pratique, vous devez commencer avec une valeur de deux et l'augmenter graduellement jusqu'à ce que le déplacement soit graduel.

Vous pouvez tirer parti d'un autre effet utile de la commande UPDATE EVERY qui est de réserver davantage de temps au Basic pour exécuter vos programmes. En utilisant judicieusement cette instruction, vous pourrez parfois accroître la vitesse de votre programme par 30%, sans supprimer la qualité graduelle de vos séquences d'animation.

Dépasser la limite des 16 objets

SYNCHRO *(Exécuter directement un programme AMAL)*

SYNCHRO [ON/OFF]

Normalement, AMOS Basic vous permettra d'exécuter 16 différents programmes AMAL à la fois. Cette limite est déterminée par la vitesse globale de l'Amiga. Chaque programme AMAL s'empare de sa tranche de temps de traitement disponible. Par conséquent, si vous utilisez le système d'interruption normal, vous disposez seulement d'un temps suffisant pour exécuter 16 programmes individuels.

La commande SYNCHRO vous permet de contourner cette restriction en exécutant vos programmes AMAL directement depuis le Basic. Au lieu d'utiliser les interruptions, tous les programmes AMAL sont maintenant lancés au moyen d'un simple appel à la commande SYNCHRO. Puisque les programmes AMAL s'éxécutent bien plus rapidement que les routines Basic équivalentes, vos animations seront toujours bien lisses. En outre, vous pourrez décider de la localisation des routines AMAL dans votre programme.

Cette commande présente un autre avantage: vous pouvez intégrer des commandes de détection de collision telles que Bob col ou Sprite Col directement dans

vos routines AMAL. Celles-ci ne sont pas disponibles depuis le système d'interruptions puisqu'elles se servent du blitter. Ce serait impossible en utilisant des interruptions.

Avant d'appeler SYNCHRO, vous devez d'abord désactiver les interruptions au moyen de SYNCHRO OFF. Il est important de faire cela **avant** de définir vos programmes AMAL, sinon vous ne serez pas autorisé à utiliser des numéros de canaux supérieurs à 15 ou vous obtiendrez un message d'erreur dans votre essai.

En raison de la puissance réelle du système d'animation, il est presque possible d'écrire des jeux d'arcade entiers en AMAL. Votre programme Basic est chargé de simples tâches telles que la gestion d'un tableau de marquage des points et le chargement de votre série d'attaques depuis le disque. Les résultats ne pourront pas être distingués en pure code machine. Cartoon Capers est un bon exemple: il est le premier jeu commercial entièrement écrit en AMOS.

Vour trouverez une démonstraton de SYNCHRO dans **Exemple 14.6** du dossier MANUEL.

Les commandes d'animation compatibles au STOS

Le STOS Basic d'origine comprenait un système d'animation puissant qui vous permettait de déplacer vos sprites selon des schémas bien complexes sous interruptions. A ce moment-là, ces commandes étaient accueillies comme un important progrès.

Bien qu'elles soient devancées par le système AMAL, elles servent de simple initiation à l'animation sur l'Amiga. Par conséquent, AMOS vous propose le système entier d'animation STOS en prime!

Si vous avez l'intention de convertir les programmes STOS en AMOS, il vous faudra noter les points suivants:

- A la différence du STOS, les schémas de déplacement en AMOS Basic peuvent être affectés à n'importe quel canal d'animation que vous souhaitez. Vous pouvez donc utiliser les commandes Move pour déplacer les bobs, les sprites ou les écrans, en vous servant exactement des mêmes techniques.

  Par défaut, tous les canaux d'animation sont affectés aux sprites machine équivalents. En pratique, vous pouvez trouver qu'il est plus facile de les remplacer par des bobs puisque ces derniers sont bien plus proches des sprites du STOS Basic. Ajoutez une séquence de commandes CHANNEL au début de votre programme comme suit:

```
Channel 1 to bob 1
Channel 2 to bob 2
: : :
```

  N'oubliez pas d'appeler DOUBLE BUFFER lors de votre procédure d'initialisation, sinon vos bobs vont scintiller d'une façon gênante lors de leur déplacement.

- Le même canal peut être utilisé pour les programmes AMAL et les animations STOS. Par conséquent, il est facile de prolonger vos programmes après les avoir converti en AMOS Basic. L'ordre d'exécution est:

AMAL
MOVE X
MOVE Y
ANIM

MOVE X *(Déplacer un sprite horizontalement)*

MOVE X n,m$

Cette commande définit une liste de déplacements horizontaux qui seront effectués ultérieurement sur un canal d'animation numéro *n*.

n peut être compris entre 0 et 15 et se rapporte à un objet que vous avez préalablement affecté au moyen de la commande CHANNEL. *m$* contient une séquence d'instructions qui déterminent la vitesse et la direction de votre objet. Ces commandes sont délimitées par des parenthèses et sont entrées sous le format suivant:

(vitesse,pas,nombre)

Aucune limite n'est imposée au nombre de commandes qu'une simple chaîne de déplacement peut comporter, mise à part la mémoire disponible.

vitesse définit une attente en 50ème de seconde entre chaque pas de déplacement successif. La vitesse peut varier entre 1 (très rapide) et 32767 (incroyablement lent).

pas spécifie le nombre de pixels du déplacement de l'objet lors de chaque opération.

Si le pas est positif, le sprite se déplacera vers la droite et s'il est négatif, il se déplacera vers la gauche.

La vitesse apparente de l'objet dépend non seulement de sa vitesse mais aussi de la grandeur du pas. Des déplacements très espacés à une vitesse modérée déplacera l'objet rapidement mais d'une façon saccadée à l'écran. Similairement, un petit pas associé à une grande vitesse déplacera également l'objet rapidement mais le déplacement sera bien plus régulier. Pour obtenir la vitesse la plus rapide, spécifiez le nombre de pas à 10 (ou -10).

nombre détermine le nombre de répétitions du déplacement. Les valeurs possibles sont comprises entre 0 et 32767. La *valeur* 0 permet d'effectuer un schéma de déplacement qui s'exécute indéfiniment.

Outre les commandes ci-dessus, vous pouvez ajouter une des instructions suivantes à la fin de votre chaîne de déplacement.

Le plus important de ces prolongements est l'instruction L (pour loop), qui se branche en amont, au début de la chaîne et redéroule toute la séquence depuis le début. Exemple:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 1,130,100,1 : Rem Définir le sprite 5
Move X 1,"(1,5,60)(1,-5,60)L"
Move On
```

L'option E vous permet d'arrêter votre objet lorsqu'il atteint un point spécifique à l'écran. Changez l'avant-dernière ligne de la séquence ci-dessus par:

```
Move X 1,"(1,5,30)E100"
```

Notez que ces points indicateurs fonctionneront seulement si l'abscisse de l'objet arrive à prendre la valeur donnée dans l'instruction. Si cette constante est mal choisie, l'objet dépassera le point indicateur et le test échouera. Exemple:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Channel 1 To Sprite 8 : Channel 2 To Sprite 10
Print At(0,5)+"Loopint OK"
Sprite 8,130,100,1
Move X 1,"(1,10,30)(1,-10,30)L"
Move On
Print At(0,10)+"Appuyez maintenant sur une touche" : Wait Key
Sprite 10,140,150,2
Move X 2,"(1,15,20)L" : Move On 2
Print At(0,15)+"Mon dieu!" :Wait Key
```

MOVE Y *(Déplacer un objet verticalement)*

MOVE Y n,m$

Cette instruction complète la commande MOVE X en vous permettant de déplacer un objet verticalement le long de l'écran. Comme auparavant, *n* se rapporte au numéro d'une séquence d'animation que vous avez affectée en utilisant la commande CHANNEL et il est compris entre 0 et 15.

m$ représente une chaîne de déplacement dont le format est identique à celui de MOVE X. Les déplacements positifs sont descendants et les déplacements négatifs sont ascendants. Exemples:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Channel 1 To Sprite 8 : Sprite 8,130,10,1 : Rem Placer le sprite
Move Y 1,"10(1,1,180)L" : Rem Itérer un sprite de 10,10 à 190,10 continuellement
Channel 2 To Screen Display 0 : Rem Affecter l'écran au canal 2
Move Y 2,"(1,4,25)(1,-4,25)" : Rem Fait monter ou descendre l'écran
Move On : Wait Key
```

MOVE ON/OFF *(Démarrer/arrêter des déplacements)*

MOVE ON/OFF [n]

Avant d'exécuter vos déplacements, il faut les activer au moyen de la commande MOVE ON.

n se rapporte à la séquence d'animation que vous souhaitez lancer et peut être

compris entre 0 et 15. Si vous l'omettez, alors tous vos déplacements seront activés simultanément.

MOVE OFF a exactement l'effet opposé: il arrête les séquences de déplacement appropriées sur leur chemin.

MOVE FREEZE *(Suspendre temporairement les déplacements de sprite)*

MOVE FREEZE [n]

La commande MOVE FREEZE suspend temporairement les déplacements d'un ou plusieurs objets à l'écran. Vous pouvez relancer ces objets au moyen de MOVE ON.

n est optionnel et spécifie le numéro du seul objet dont le déplacement sera suspendu par cette instruction.

MOVON *(Renvoyer l'état de déplacement)*

x=MOVON(n)

Cette commande vérifie si un objet est en cours de déplacement par les instructions MOVE X et MOVE Y. Elle renvoie -1 (vrai) si l'objet *n* est en cours de déplacement et 0 (faux) si sa position est stationnaire. Ne confondez pas cette commande avec MOVE ON.

Notez que MOVE ON recherche seulement les déplacements produits en utilisant les commandes MOVE. Elle ne détectera pas les animations produites par AMAL.

ANIM *(Animer un objet)*

ANIM n,a$

Anim déplace automatiquement un objet dans une séquence d'images en créant un effet d'animation progressif à l'écran. Ces animations sont effectuées 50 fois par seconde en utilisant des interruptions pour qu'elles puissent être exécutées en même temps que vos programmes Basic.

n est le numéro de canal qui spécifie le sprite ou le bob qui doit être animé par cette instruction.

a$ contient une série d'instructions qui définit votre séquence d'animation. Chaque opération est divisée en deux éléments individuels délimités par des parenthèses.

image est le numéro de l'image devant être affichée lors de l'affichage de chaque image de l'animation.

durée spécifie la durée de représentation de cette image à l'écran (en 50ème de seconde). Exemple:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Channel 1 To Sprite 8 : Sprite 8,200,100,1
Anim 1,"(1,10)(2,10)(3,10)(4,10)"
```

Anim on : Wait key

Comme avec l'instruction MOVE, il existe également une instruction L vous permettant de répéter vos animations de façon continue. Il vous suffit de changer la commande ANIM de l'exemple précédent par celle qui suit:

Anim 1"(1,10)(2,10)(3,10)(4,10)L"

ANIM ON/OFF *(Lancer une animation)*

ANIM ON/OFF[n]

Cette commande active une série d'animations préalablement créée au moyen de la commande ANIM. *n* spécifie le numéro de la séquence d'animation à initialiser. Si vous l'omettez, toutes les séquences courantes d'animation seront immédiatement lancées.

ANIM OFF[n]
Cette commande interrompt une ou plusieurs séquences d'animation lancées par ANIM ON.

ANIM FREEZE *(Figer une animation)*

ANIM FREEZE[n]

Cette commande fige temporairement la séquence d'animation courante à l'écran. *n* sélectionne une seule séquence d'animation dont l'exécution doit être suspendue. Si vous omettez de le spécifier, toutes les animations courantes seront affectées. Mais vous pouvez les relancer à votre gré par un simple appel à l'instruction ANIM ON.

15 Les graphiques en arrière-plan

De nos jours, il n'est pas rare qu'un jeu d'arcade contienne des centaines d'écrans différents. Grâce au compactage, il est possible de faire tenir un écran de 32 couleurs dans 30k de mémoire. Ainsi, 100 écrans seraient l'équivalent de 3 mégaoctets de données. Imaginez la difficulté que vous auriez à les faire entrer dans un A500 standard!

La façon classique de contourner cette restriction est d'élaborer vos arrière-plans à partir d'un jeu de simples blocs fonctionnels. Une fois que ces tuiles ont été créées, elles peuvent être placées sur l'écran dans l'ordre que vous voulez. Ainsi, le même jeu de tuiles peut être réutilisé pour produire un très grand nombre d'écrans potentiels. Chaque écran est alors mémorisé sous la forme d'un simple bloc composé de ses modules et il demande seulement une portion minuscule de la mémoire d'origine.

Afin d'exploiter ce système, vous aurez évidemment besoin d'une méthode de définition de vos cartes graphiques. Comme vous l'avez pu deviné, nous avons prévu à cet effet un utilitaire de définition de cartes sur la disquette programme AMOS. Vous trouverez toutes les explications dans le fichier documentation l'accompagnant.

AMOS Basic comprend également un certain nombre d'instructions spéciales concernant le tracé de vos tuiles à l'écran. Celles-ci facilitent la création d'arrière-plans au scrolling rapide qui authentifient un jeu d'arcade moderne.

Les icônes

Les icônes sont des images indépendantes particulièrement conçues pour la création d'écrans en arrière-plan. Une fois tracé, l'icône est fixé en permanence à l'écran. Par conséquent, vous ne pouvez pas le changer de position en utilisant le système d'animation AMAL.

Toutes les icônes sont rangées dans leur propre banque de mémoire d'AMOS (banque 2). Cette banque est créée au moyen de l'utilitaire de définition de Sprites (se trouvant sur la disquette programme AMOS) et sera automatiquement sauvegardé avec vos programmes Basic.

Comme les bobs, les icônes sont affichées en utilisant le blitter de l'Amiga. Mais puisque les icônes sont essentiellement des objets statiques, ils sont habituellement retracés dans le mode REPLACE. Vos icônes effaceront donc tous les graphiques existant à la position courante de l'écran.

PASTE ICON *(Tracer une icône)*

PASTE ICON x,y,n

Cette commande trace une icône numéro n aux coordonnées GRAPHIQUES *x,y* de l'écran. *n* est le numéro de l'icône devant être affichée. Celle-ci doit avoir été préalablement rangée dans la banque ICONES (2).

Les icônes peuvent être facilement positionnées à n'importe quel point de l'écran, mais elles sont soumises aux règles habituelles de découpage. Exemple:

```
Load "AMOS_DATA : Icons/Map_icons.abk"
Screen Open 0,320,256,32,Lowres : Cls 0 : Get Icon Palette
Rem Tracer ligne de contour d'écran
For X=1 To 11 : Paste Icon X*32,0,1 : Next X
For Y=1 To 7 : Paste Icon 0,Y*32,1 : Paste Icon 288,Y*32,1 : Next Y
For X=1 To 11 : paste Icon X*32,223,1 : Next X
```

Notez que si vous utilisez le double tamponnage, vos icônes seront retracées sur les écrans logique et physique. Puisque cette opération est plutôt lente, précédez-la d'un appel à AUTOBACK 0 avant de tracer vos icônes à l'écran. Ceci limite vos icônes à l'écran logique courant. Vous pouvez alors copier tout l'arrière-plan directement sur l'écran physique au moyen de SCREEN COPY, et économiser ainsi un temps considérable.

Si vous souhaitez étudier un autre exemple, référez-vous au programme MAPVIEW de la disquette de données d'AMOS. Il affiche l'écran en arrière-plan que vous avez créé en utilisant l'éditeur de cartes AMOS.

GET ICON *(Créer une icône)*

GET ICON [s,] i,tx,ty TO bx,by

Cette commande saisit une image depuis l'écran et la charge dans l'icône *i*. Si cette icône n'existe pas, elle sera créée pour vous dans la banque 2. Cette banque sera automatiquement réservée par le système si besoin.

i est le numéro de votre icône, à partir de 1. *tx,ty* et *bx,by* définissent les angles supérieurs et inférieurs d'une zone rectangulaire renfermant la région sélectionnée.

s détermine le numéro de l'écran qui sera utilisé comme la source de votre image. Si vous l'omettez, l'image sera saisie depuis l'écran courant. Exemple:

```
Erase 2
F$=Fsel$("*.*","","Load a screen) : If F$=""Then Direct
If Exist(f$) Then Load Iff F$,0 Else Direct
SH=Screen Height : H=SH/32-1 : Sw=Screen Width : W=SW/32-1
Z=1
For Y=0 To H
   For X=0 To W
      Rem Saisir une icône
      Get Icon Z,X*W+1,Y*32 To X*W+31,Y*32+31
   Next X
Next Y
Cls 0
Do
   Paste Icon Rnd(Sw-1),Rnd(SH-1),Rnd(H*W)+1
Loop
```

GET ICON PALETTE *(Saisir les couleurs d'icônes)*

GET ICON PALETTE
Cette commande saisit les couleurs des images d'icônes dans la banque 2 et les charge dans la palette de l'écran courant. Cette commande est normalement utilisée pour initialiser l'écran après avoir chargé des icônes de la disquette. Exemple:

```
Rem Charger des icônes de la disquette AMOS DATA
Load "AMOS_DATA:Icons/Map_icons.abk"
Get Icon Palette
Paste Icon 100,100,1
```

DEL ICON *(Effacer des icônes)*

DEL ICON n[TO m]

Cette commande efface une ou plusieurs icônes de la banque d'icônes. *n* est le numéro de la première icône devant être supprimée.

m est le numéro optionnel de la dernière icône de la liste devant être effacée. Si vous le définissez, toutes les icônes du *premier* au *dernier* seront effacées l'une après l'autre. Exemple:

```
Load "AMOS_DATA:Icons/Map_icons.abk"
Paste Icon 100,100,1
Del Icon 1
Wait Key
Paste Icon 100,100,1
```

Lorsque la dernière icône d'une banque a été effacée, toute la banque sera supprimée de la mémoire.

MAKE ICON MASK *(Rendre la couleur zéro transparente)*

MAKE ICON MASK [n]

Normalement, les icônes que vous tracez à l'écran remplaceront entièrement l'arrière-plan existant. L'icône semblera s'afficher dans un rectangle rempli de la couleur zéro.

Si vous voulez éviter cet effet et appliquer vos icônes sur les graphiques courants, il faudra créer un masque pour vos icônes. Ce masque informe AMOS que la couleur zéro doit être considérée comme transparente.

n est le numéro de l'icône devant être affecté. Si vous l'omettez, un masque sera défini pour toutes les icônes de la banque.

Vous trouverez une démonstration de cet effet dans le fichier **EXEMPLE 15.1** du dossier MANUEL.

Les blocs d'écran

AMOS Basic vous livre un jeu de commandes BLOCK puissantes vous permettant de saisir le bloc d'une image en mémoire et de le coller à n'importe quel endroit de l'écran. Ces instructions sont principalement utilisées pour contenir des données temporaires, puisque vos blocs ne peuvent pas être sauvegardés avec vos programmes Basic.

Les blocs sont particulièrement utiles dans la construction de boîtes de dialogue, puisqu'ils peuvent être utilisés pour sauvegarder les zones en arrière-plan avant d'afficher de nouveaux graphiques.

Ils peuvent servir aux jeux de patience comme Doubles Personnalités, chaque bloc de l'écran recevant une seule portion de votre image. Vous pouvez ensuite brouiller vos images en réarrangeant les blocs à l'écran avec PUT BLOCK.

GET BLOCK *(Saisir un bloc d'écran en mémoire)*

GET BLOCK n,tx,ty,w,h[,mask]

Cette commande saisit une zone rectangulaire dans le bloc numéro *n*, en partant des coordonnées *tx,ty*.

n est le numéro du bloc compris entre 1 et 65535. *tx,ty* représentent les coordonnées de l'angle supérieur gauche de votre bloc. *w,h* représentent respectivement la largeur et la hauteur de ce bloc.

mask est un indicateur qui choisit si un masque doit être créé pour votre nouveau bloc.

mask=0	Mode de remplacement. Si ie bloc est tracé à l'écran, il détruira les graphiques à la position courante.
mask=1	Calcule un masque pour le bloc. La couleur zéro est alors considérée comme transparente.

PUT BLOCK *(Copier un bloc précédemment créé à l'écran)*

PUT BLOCK n[,x,y]
PUT BLOCK n,x,y,planes[,minterms]

Cette commande copie le bloc numéro n sur l'écran. *x,y* spécifient la position du nouveau bloc à l'écran. Si vous omettez ces paramètres, le bloc sera retracé aux coordonnées de l'écran d'origine.

Notez que toutes les opérations de tracé seront découpées pour entrer dans l'écran courant, depuis la frontière la plus proche de 16 pixels.

Vous trouverez une démonstration des commandes BLOCK dans **le fichier EXEMPLE 15.2.** Nous avons également prévu deux ou trois autres commandes optionnelles pour les programmeurs confirmés. Ces dernières ne sont pas nécessaires pour la plupart des applications, seulement si vous voulez créer des effets spéciaux et étranges!

planes représente un mode point définissant la palette de couleurs qui sera utilisée

pour le remplissage de votre bloc. L'écran de l'Amiga est divisé en segments connus sous le nom de plans binaires.

Chaque plan comprend un seul bit pour chaque point de l'écran de l'Amiga. Si le hardware de l'Amiga affiche ce point, elle combine les bits de chaque plan pour calculer le numéro de couleur demandé. Chaque bit de *planes* représente l'état d'un seul plan binaire. S'il est forcé à un, alors le plan sélectionné sera tracé par cette instruction, sinon il sera complètement ignoré. Le premier plan est représenté par le bit zéro, le deuxième par le bit un, etc...

D'habitude, le bloc est affiché dans tous les plans binaires disponibles. Il correspond à la configuration binaire de %111111.

minterm sélectionne le mode de manipulation d'objet utilisé pour copier votre bloc à l'écran. Vous trouverez une description complète des modes possibles de tracé dans le paragraphe sur SCREEN COPY.

La meilleure façon d'apprendre à utiliser ces options est de les mettre en pratique!

DEL BLOCK *(Effacer un bloc d'écran)*

DEL BLOCK n

Cette commande efface un ou plusieurs blocs et libère donc de la mémoire pour l'AMOS Basic.

DEL BLOCK	Efface **tous** les blocs courants
DEL BLOCK n	Efface le bloc numéo n.

GET CBLOCK *(Sauvegarder et compacter une image d'écran)*

GET CBLOCK n,x,y,sx,sy

La commande GET CBLOCK sauvegarde et compacte une zone rectangulaire de l'écran. Le système de compactage utilisé par cette commande a été spécialement optimisé pour être rapide. Ainsi, elle est loin d'être aussi efficace que les routines spécialisées de compactage AMOS proposées par les instructions PACK ou SPACK.

CBLOCKS sont souvent utilisées pour saisir la zone se trouvant au dessous de vos boîtes de dialogue. Une fois le dialogue terminé, l'écran peut reprendre rapidement son état d'origine. Référez-vous au fichier **EXEMPLE 15.3** du dossier MANUEL pour étudier une démonstration.

n spécifie le numéro de votre bloc et peut être compris entre 1 et 65535.

x/y représentent les coordonnées de l'angle supérieur gauche du bloc. La largeur de votre bloc exprimée dans w est toujours arrondie à un multiple de huit.

PUT CBLOCK *(Afficher un bloc créé au moyen de CBLOCK)*

PUT CBLOCK n [,x,y]

Cette commande place le bloc *n* aux coordonnées *x,y* de l'écran courant. Si vous

omettez de spécifier les coordonnées destinataires, le bloc sera retracé à la position de son écran d'origine. Notez également que *x* est automatiquement arrondi à la frontière de huit pixels la plus proche.

DEL CBLOCK *(Effacer un bloc d'écran défini à l'aide de GET CBLOCK)*

DEL CBLOCK [n]

Cette commande efface tous les blocs de la mémoire. Si *n* est spécifié, seul le bloc *n* sera effacé.

16 Menus

Si vous avez utilisé l'Amiga pendant quelques temps, vous serez sans doute à l'aise avec les menus. Bien que cela puisse vous paraître impossible, AMOS s'est basé sur le système existant et l'a amélioré jusqu'à ce qu'il devienne presque méconnaissable.

Vous pouvez créer les menus sur huit niveaux et vous pouvez repositionner chaque article particulier du menu à l'écran à votre gré. Vous pouvez imprimer les titres du menu dans n'importe quelle combinaison de styles et de couleurs et faire apparaître des bobs ou des icônes dans vos menus en utilisant un spectaculaire langage de définition de menu.

AMOS Basic est également impressionnant pour ce qui est de gérer un menu. Une commande ON MENU fonctionnant par interruption est prévue et elle peut automatiquement se brancher à un point sélectionné de votre programme en fonction de l'option choisie. En outre, vous pouvez accéder directement à toutes les options du menu depuis le clavier en utilisant l'instruction MENU KEY.

Chargez le programme **EXEMPLE 16.1** du dossier MANUEL qui vous donne une démonstration des effets stupéfiants que vous pouvez obtenir avec ce système. Vous aurez du mal à croire à la puissance réelle de ces commandes avant de les voir en action!

Comment utiliser un menu

Tous les menus d'AMOS sont appelés en maintenant le bouton droit de la souris abaissé comme de coutume. Dès que vous avez activé un menu, vous pouvez sélectionner une option directement avec le curseur de la souris. Si vous relâchez le bouton, le numéro d'option que vous avez choisi sera renvoyé au programme.

Vous pouvez repositionner les menus en plaçant le curseur de la souris sur l'angle supérieur **gauche** d'un article et en maintenant le bouton gauche abaissé. Une petite boîte apparaîtra sur la barre du menu que vous pouvez déplacer au travers de l'écran au moyen de la souris.

En outre, en abaissant la touche SHIFT, vous figerez le menu, ce qui vous permet de l'explorer sans sélectionner une des options proposées. Vous pouvez également utiliser une des fonctions de la souris telles qu'un ralentissement ou une sélection d'axe en conjonction avec vos menus.

Créer un menu simple

Vous pouvez créer les menus d'AMOS directement dans vos programmes ou au moyen d'un définisseur spécial de menu prévu sur la disquette de programme AMOS.

Si vous n'avez jamais utilisé de menus, la simple variété des commandes de menus qui sont à votre disposition peut vous dépasser un peu. Voici une brève description de quelques fonctions Basic vous permettant de vous initier aux menus AMOS sans vous donner trop de mal.

Définir la ligne de titre

Dans la création d'un menu, la première étape est de définir la *ligne de titre* d'un menu. Vous utilisez pour cela la commande MENU$. Sa syntaxe la plus simple est la suivante:

MENU$ *(Définir le titre du menu)*

MENU$(n)=titre$

Cette commande crée une ligne de titre pour votre menu. Chaque en-tête est affectée de son propre numéro, commençant par un et augmentant de gauche à droite. Ainsi, le titre tout à gauche est représenté par un, le suivant par deux, etc...

Le texte de *titre$* représente le nom de l'option affiché au nouveau menu. Voici un simple exemple construisant une ligne de menu comprenant simplement deux titres: ACTION et SOURIS.

```
Menu$ (1)=" Action "
Menu $ (2)=" Souris "
```

Un espace est prévu après *Action* pour séparer ce titre de *Souris*, le menu suivant. Vous devez maintenant spécifier une liste d'options à associer à vos nouvelles en-têtes. Celles-ci forment une barre verticale qui se mettra en place lors de la sélection d'un titre par la souris.

MENU$(t,o) *(Définir une option menu)*

MENU$(t,o)=option$

Cette deuxième syntaxe de MENU$ définit une liste d'options qui seront affichées dans la barre du menu.

t est le numéro de l'en-tête du menu sous laquelle votre option apparaîtra. *o* est le numéro d'options que vous souhaitez installer dans la barre du menu. Toutes les options sont numérotées en partant de la plus haute, à laquelle le chiffre un est attribuée.

La seule limitation physique imposée à la taille de votre menu est la capacité de la mémoire, mais il est sage de vous limiter à moins de dix options par titre afin que la complexité de vos menus soit convenable.

option$ représente le nom de votre nouvelle option. Celle-ci peut se composer de n'importe quel libellé que vous souhaitez. Par exemple, essayez d'ajouter les lignes suivantes au programme ci-dessus:

```
Rem Menu d'intervention
Menu$ (1,1)= " Sortir "
Rem Menu de la souris
Menu$ (2,1)=" Flèche "
Menu$ (2,2)=" Curseur "
Menu$ (2,3)=" Horloge "
Wait Key
```

Cette instruction spécifie une liste d'autres commandes pour les menus ACTION et SOURIS. Si vous essayez de faire fonctionner ce programme comme il se présente, rien

ne se produira parce que les menus doivent être initialisés par un appel à la commande MENU ON. Entrez cette dernière dans le programme ci-dessus avant l'instruction "Wait Key". Faites fonctionner maintenant le programme exemple et sélectionnez les articles du menu au moyen du curseur de la souris. Rappelez-vous de maintenir d'abord le bouton DROIT de la souris abaissé!

MENU ON *(Activer le menu)*

MENU ON

Cette commande active un menu défini au moyen de la commande MENU$. La ligne du menu apparaîtra alors automatiquement lorsque vous appuyez sur le bouton DROIT de la souris. Pour lancer le menu précédant, insérez la ligne suivante après les instructions de définition.

```
Menu On
```

Rendez-vous à la fenêtre Directe et jouez avec les menus. Sélectionnez les options en appuyant sur le bouton droit de la souris.

Lire un menu simple

Dès que vous avez créé votre menu et activé le système de menu d'AMOS, il vous faudra trouver les options que vous avez sélectionnées. Pour cela, utilisez la simple syntaxe de la commande CHOICE.

=CHOICE *(Lire un menu)*

selected=CHOICE(1)

Cette commande renvoie une valeur de -1 (vrai) si le menu est mis en surbrillance. De la même manière, vous pouvez retrouver le numéro d'option qui a été choisi au moyen d'un paramètre d'une valeur de deux.

```
item=CHOICE(2)
```

Essayez d'ajouter les lignes suivantes à l'exemple précédent:

```
Do
   Rem If Choice=-1 peut être simplifié à: If Choice, comme ci-dessous:
   If Choice and Choice(1)=1 Then Exit
   If Choice(1)=2 and Choice(2)<>0 Then Change Mouse Choice(2)
Loop
```

Cette commande change la forme du curseur de la souris en fonction de l'option que vous avez choisi au menu. Vous trouverez une démonstration complète de ces menus dans le fichier Exemple 16.2 du dossier MANUEL.

Les fonctions de menu perfectionnées

Nous allons maintenant étudier certaines des fonctions de menu les plus avancées qui sont proposées dans l'AMOS Basic. Lorsqu'elles sont utilisées correctement, ces menus AMOS peuvent ajouter une toute nouvelle dimension à vos programmes.

MENU$ *(Créer un menu)*

MENU$(.,)=normal$[,selectionné$][,inactif$][,arrière-plan$]

Cette commande définit l'apparence de chaque article d'un menu. A la différence des menus classiques d'Amiga, ces articles ne sont pas limités au texte normal. Ils peuvent aussi comprendre des commandes intégrées vous permettant de tracer des bobs, des icônes ou des graphiques à n'importe quel point de la ligne de menu.

Vous pouvez omettre n'importe quel paramètre de cette instruction pour pouvoir changer chaque partie de la description d'un menu. L'apposition des signes "" dans la chaîne de votre menu **effacera** la définition existante. Similairement, vous pouvez garder la valeur d'origine en la délimitant par des virgules. Par exemple:

```
Menu$(1)=" Action ","" : Rem Effacer la deuxième option
Menu$(2)=" Souris 2 ",,"" : Rem Modifier le titre sans rien changer d'autre
```

La position de l'article du menu au menu est indiquée par une liste d'au plus huit paramètres séparares par des virgules. La syntaxe générale est la suivante:

(article)/(article,option)/(article,option,sous-option)...

normal$* est une chaîne définissant l'apparence normale d'un article lors de son affichage au menu. *selectionné$ change l'effet de surbrillance d'une option de menu sélectionnée à l'aide de la souris. Par défaut, les articles sélectionnés apparaissent en inversé.

Inactif$ change l'apparence d'un article qui a été désactivé au moyen de la commande MENU INACTIVE.

arrière-plan$ crée un arrière-plan pour les articles de votre menu lors de leur tracé initial. Celui-ci aura généralement la forme d'une boîte quelconque créée au moyen des commandes internes ligne ou de Barre.

A partir de maintenant, nous abrégeons ces paramètres en utilisant une notation standard:

position$=[,selectionné$][,inactif$][,arrière-plan$]

L'arborescence du menu

Le niveau d'un article au menu est déterminé par sa position dans *l'arborescence du menu*.

```
Menu$(1)="Titre"
Menu$(1,1)="Option 1"
```

```
Menu$(1,2)="Option 2"
Menu$(1,2,1)="Article 1"
```

Cette instruction définit un menu simple. La structure d'un menu est similaire à celle d'un tableau. Chaque niveau du menu est représenté par sa propre dimension dans le tableau et il est contrôlé en utilisant une variante de la commande MENU$.

Le premier niveau représente la *ligne de titre* qui apparaît en haut de vos menus. Il peut être défini au moyen de la commande suivante:

```
menu$(n)=titre$[positions$]
```

n correspond à la position du titre à la gauche de l'écran et position$ se rapporte aux trois chaînes optionnelles définissant l'apparence générale du menu. Il est important de définir d'abord le titre de vos menus puisque celui-ci **dimensionne** le tableau. Tous les autres articles peuvent être créés dans l'ordre que vous souhaitez.

Chaque article est associé à une liste d'options qui sont visualisées à la sélection du menu. Celles-ci constituent le deuxième niveau de la structure du menu et sont définies en utilisant une deuxième variante de la commande MENU$.

```
Menu$(n,option)=Article$[position$]
```

option représente le numéro de l'article déterminé depuis la gauche et le haut de la barre du menu. Aucune limite, autre que la capacité de la mémoire, n'est imposée au nombre d'options pouvant être reliées à un seul titre.

En retour, chaque option peut être associée à ses propres sous-menus comportant eux-mêmes huit niveaux.

```
Menu$(n,option,sous-option)=Article$[position$]
```

Un menu peut être prolongé ou modifié à n'importe quel point de votre programme. Ne changez jamais l'écran courant lors de la création d'un menu car cela provoquerait un message d'erreur.

Afin de vous familiariser avec l'arborescence du menu, chargez le programme **EXEMPLE 16.3** du dossier MANUEL.

=CHOICE (Lire un menu)

```
item=CHOICE[(dimension)]
```

La fonction CHOICE vous permet de vérifier si vous avez mis une option en surbrillance au menu courant. Si vous avez sélectionné un article (au niveau le plus bas), CHOICE renverra une valeur de -1 (vrai), sinon elle sera de 0 (faux). Après avoir appelé cette fonction, l'état du menu sera automatiquement remis sur 0 (faux). Cela permet de ne pas sélectionner un seul menu plusieurs fois par erreur.

La deuxième variante de la commande CHOICE renvoie l'option sélectionnée au niveau demandé.

article=CHOICE(dimension)
dimension indique le niveau du menu qui doit être lu. Comme vous savez, un niveau numéro 1 correspond à *la ligne de titre* du menu. Similairement, les niveaux se trouvant entre 2 et 8 indiquent le numéro d'une option qui a été choisie. Si l'article d'un menu n'a pas été sélectionné, *article* sera chargé d'une valeur de zéro. Par exemple:

```
Menu$(1)="Menu"
Menu$(1,1)="Option 1"
Menu$(1,2)="Option 2"
Menu$(1,2,1)="Option 2.1"
Menu on
Do
   If Choice Then Print Choice (1),Choice(2),Choice(3)
Loop
```

Si vous vouliez mettre en place des menus plus grands avec ce système, il vous faudrait utiliser une longue liste de IF...THEN dans votre programme pour traiter chaque possibilité. Cela entraînerait un petit retard important dans votre programme pendant la lecture des menus. Il vous serait également très difficile de modifier votre programme par la suite.

Heureusement, AMOS Basic vous propose ume méthode vous permettant de gérer les plus grands menus sans vous donner de mal.

ON MENU PROC *(Sélection automatique de menu)*

ON MENU PROC proc1[,proc2...]

Chaque titre de votre menu peut être affecté de sa propre procédure qui s'exécutera automatiquement à la sélection d'une option. L'action de cette commande est similaire aux lignes suivantes du langage d'AMOS Basic:

```
If CHOICE
   If CHOICE(1)=1
      Proc1
   Endif If
   CHOICE(1)=2
      Proc2
   Endif
: : :
: : :
Endif
```

Il existe toutefois une différence importante entre la commande ON MENU et les instructions ci-dessus. ON MENU est effectuée 50 fois par seconde au moyen d'interruptions et n'affecte pas l'ensemble du déroulement de votre programme. Cela signifie que votre programme peut être en train de faire quelque chose de complètement

différent pendant que les menus sont vérifiés par le système.

Si vous sélectionnez un article au menu, la procédure demandée s'exécutera immédiatement sans avoir aucune autre intervention sur partie de votre programme en question. Votre procédure peut alors utiliser la commande CHOICE pour trouver l'option qui a été choisie et effectuer l'action désirée.

A l'achèvement de cette procédure, votre programme reviendra à l'instruction sur l'appel à ON MENU. Voici un exemple:

```
Menu$(1)="Action" : Menu$(1,1)="Nombre" : Menu$(1,2)="Sortir"
Menu on : Rem activer menu
On Menu Proc ACTION
On Menu On : Rem Activez commande du menu
Do:rem Tapez quelques caractères
   X$=Inkey$ : If x$<>"" Then Print X$; : Inc W
Loop
Procedure ACTION
   Shared W
   If Choice(2)=1
        Locate 0,0 : Print "Vous avez tapé la lettre ";W;" : W=0
        On Menu On : Rem initialiser les menus
   Endif
   If CHOICE(2)=2 Then Edit
End Proc
```

Deux ou trois points sont à noter dans cet exemple. Tout d'abord, remarquez que la séquence ON MENU est activée au moyen de la commande ON MENU ON. Cette commande **doit** être appelée après l'exécution de la procédure de gestion du menu puisqu'elle est nécessaire pour relancer le système de menu. Notez également l'utilisation de INKEY$ au lieu de INPUT. La commande INPUT suspend les recherches du menu pendant que vous entrez une ligne. Vous pouvez utiliser toutes les autres commandes sans problèmes, y compris WAIT, WAIT VBL et WAIT KEY. Référez-vous au programme **EXEMPLE 16.4.**

ON MENU GOSUB *(Sélection automatique du menu)*

ON MENU GOSUB label1[, label2,...]

Cette commande lance une des sous-routines en fonction de l'option que vous avez sélectionnée. Après avoir appelé cette commande et créé vos sous-routines, les menus seront contrôlés automatiquement 50 fois par seconde.

Notez que chaque titre de la ligne du menu est géré par sa propre sous-routine. Celle-ci est différente de son équivalente en AMIGA Basic qui contrôle tout le menu au moyen d'une simple routine.

Vous devez ensuite activer le système de menu par un appel à l'instruction ON MENU. Les menus doivent être initialisés de cette manière avant de se rebrancher en amont du programme principal avec RETOUR. Notez également que *label* **ne peut pas** être remplacé par une expression puisque l'étiquette sera seulement évaluée une seule

fois lors du lancement du programme. Voir ON MENU PROC et ON MENU ON/OFF.

ON MENU GOTO *(Sélection automatique du menu)*

ON MENU GOTO label 1[,label 2,...]

Cette commande a été remplacée par les instructions plus puissantes ON MENU PROC et ON MENU GOSUB. Elle a été conçue dans un but de compatibilité avec les programmes écrits en STOS Basic. A la sélection d'un menu, le programme se branchera sur l'étiquette appropriée. Voir ON MENU PROC, ON MENU GOSUB.

ON MENU ON/OFF *(Activer/Désactiver la sélection automatique du menu)*

ON MENU ON

Cette commande active le système automatique de menu créé par les commandes ON MENU PROC/GOSUB/GOTO. Après qu'une sous-routine ait été sollicitée de cette façon, le système sera **désactivé**. Ainsi, il est important de réactiver le système à l'aide de ON MENU ON avant de revenir au programme principal.

ON MENU OFF

Cette commande fige le système automatique de menu. Elle est utile lorsque votre programme exécute une procédure qui doit être effectuée sans interruptions, telle que le chargement et la sauvegarde sur disquette. Vous pouvez réactiver les menus au moyen de ON MENU ON.

ON MENU DEL *(Effacer les étiquettes utilisées par ON MENU)*

ON MENU DEL

Cette commande efface la liste interne d'étiquettes ou de procédures créées par les commandes ON MENU. Vous pouvez réorienter vos menus vers une autre partie de votre programme au moyen d'un autre appel à ON MENU. **Attention!** Utilisez seulement cette commande après avoir désactivé les menus à l'aide de ON MENU OFF.

Les raccourcis clavier

En dépit de l'attrait incontestable des menus, certains utilisateurs préfèrent appeler les options d'un programme directement depuis le clavier. Bien que les menus soient certainement faciles pour les débutants, l'appel d'une option depuis le clavier peut être bien plus rapide, une fois que vous vous êtes familiarisé avec un programme.

AMOS Basic vous permet d'affecter un raccourci clavier à n'importe quel des articles de votre menu. La frappe de ces touches est interprétée exactement de la même façon que l'accès aux options équivalentes du menu. Vous pouvez les utiliser à l'aide des commandes du menu d'AMOS Basic, y compris avec ON MENU.

MENU KEY

(Affecter une touche à un article du menu)

MENU KEY (,,) TO c$
MENU KEY (,,) TO scan[,shift]

Cette commande vous permet d'affecter une touche à n'importe quel article d'un menu précédemment défini. La seule restriction imposée est que l'article que vous avez spécifié **doit** se trouver au niveau inférieur de votre menu. Ainsi, vous ne pouvez pas utiliser un raccourci clavier pour sélectionner un sous-menu puisque chaque commande doit correspondre à une seule option du menu.

c$ est une chaîne contenant un seul caractère à affecter à l'option du menu. Tous les caractères supplémentaires de la chaîne seront rejetés.

Chaque touche du clavier de l'Amiga est affecté de son propre *code de position*. En utilisant ce code, vous pouvez affecter les touches n'ayant pas d'équivalents en Ascii à un menu. Voici une liste de codes de position de touches pouvant être utilisées avec vos menus.

Code de position	Touches
80-89	Touches de fonction F1 à F10
95	Help
69	Esc

shift est un mode point optionnel vous permettant de rechercher les combinaisons des touches de commande telles que Alt+Help ou Control+D. La syntaxe de *shift* est la suivante:

Bit	Touche testée	Notes
0	Touche Shift	Une seule touche de gauche peut être de gauche testée à la fois
1	Touche Shift de droite	
2	Majuscules	Soit ON ou OFF
3	Contrôle (Cntrl)	
4	Alt de gauche	
5	Alt de droite	
6	Amiga de gauche	C'est la touche Commodore sur certains claviers
7	Amiga de droite	

Notez que si vous définissez plus d'un seul bit dans cette configuration, il vous faudra appuyer sur plusieurs touches simultanément pour appeler l'article de votre menu. Vous pouvez désactiver tous ces raccourcis clavier à l'aide de MENU KEY sans paramètre. Par exemple:

```
Menu Key(1,10):rem Désactive le circuit clavier affecté à l'article (1,10)
```

L'ajout de raccourcis clavier à un menu à l'aide de la commande MENU KEY est une opération banale; nous vous conseillons fortement de les intégrer telles quelles dans vos programmes. Voici un exemple recherchant les 10 touches de fonction de l'Amiga:

```
Menu$(1)=" Touches de fonction"
For A=1 To 10
   OPT$="F"+Str$(A)+""
   Menu$(1,A)=OPT$
   Menu Key(1,A) To 79+A
Next A
Menu On
Do
   If Choice Then Print "Vous venez d'appuyer sur une touche de fonction";Choice(2)
Loop
```

Les commandes de contrôle du menu

MENU ON *(Activer un menu)*

MENU ON [banque]

Cette commande active un menu qui a été précédemment défini dans votre programme. Le menu s'affichera sur l'appui du bouton droit de la souris et vous pourrez sélectionner les options de la façon habituelle. Si un numéro de banque est inclu dans l'instruction, alors le menu sera saisi depuis la banque concernée. Référez-vous à MAKE MENU BANK pour plus de précisions.

MENU OFF *(Désactiver temporairement un menu)*

MENU OFF

Cette commande est exactement l'opposé de MENU ON. Elle fige temporairement l'action du menu tout entier. Vous pouvez relancer le menu à n'importe quel moment au moyen de MENU ON.

MENU DEL *(Effacer un ou plusieurs articles au menu)*

Cette commande efface le menu sélectionné de la mémoire de l'Amiga et libère l'espace créé pour le reste de votre programme. Il existe deux syntaxes possibles de cette commande.

MENU DEL

Efface le menu **en entier**. **Attention!** Cette commande est irrévocable!

MENU DEL(,,)

Efface simplement une portion du menu. Les paramètres (,,) peuvent contenir une liste de huit valeurs séparées par des virgules. Celles- ci indiquent la position précise de l'article dans l'aborescence du menu. Par exemple:

```
Menu Del(1) : Rem Effacer le titre numéro 1
Menu Del(1,2) : Rem Supprimer l'option 2 du titre 1
```

MENU TO BANK *(Sauvegarder les définitions du menu dans une banque de mémoire)*

MENU TO BANK n

Cette instruction vous permet de sauvegarder tout un arbre de menus dans la banque de mémoire *n*. Si la banque *n* existe déjà, vous obtiendrez un message d'erreur *banque déjà réservée*.

Après avoir enregistré un menu de cette façon, il sera automatiquement sauvegardé avec votre programme Basic. En conservant les définitions de votre menu dans une banque de mémoire, vous pouvez réduire considérablement la taille des listings de vos programmes. Cela libèrera un espace précieux dans la mémoire des éditeurs et vous permettra d'écrire de plus longs programmes Basic en utilisant exactement la même place en mémoire.

BANK TO MENU *(Restaurer une définition de menu sauvegardée dans une banque de menu)*

BANK TO MENU n

Cette instruction permet de donner la définition d'un menu en utilisant les données du menu de la banque numéro *n*. Votre menu sera exactement reconstitué dans l'état dans lequel il a été initialement sauvegardé. Si le menu est complexe, ce processus peut prendre un peu de temps. Pour activer votre nouveau menu, appelez l'instruction MENU ON.

MENU CALC *(Recalculer un menu)*

MENU CALC

Une des fonctions les plus attractives des menus d'AMOS est la facilité avec laquelle on peut les modifier pendant le cours d'un programme. Après avoir créé votre définition initiale, vous pouvez ajouter de nouveaux articles et remplacer les options existantes à volonté.

Tous les articles de votre menu sont automatiquement repositionnés lors de la sélection du menu par le bouton droit de la souris. Cela prendra un peu de temps, si vos menus sont très longs. MENU CALC vous permet d'effectuer ce processus au point le

plus approprié de votre programme, en évitant ainsi des attentes inutiles et indésirables.

Pour arrêter l'appel d'un menu alors qu'il est en cours de modification, nous vous conseillons de figer les menus avec MENU OFF au début de votre procédure. Vous pouvez alors relancer le menu sans inquiétude au moyen de la commande MENU ON.

Les menus élaborés sont particulièrement utiles pour les jeux d'aventure puisque chaque lieu (dans le jeu) peut avoir ses propres options de menu pouvant être actualisées en fonction du jeu du joueur.

Les commandes intégrées du menu

Sur option, chaque chaîne du menu peut comprendre un jeu puissant de commandes intégrées vous permettant de personnaliser à l'extrême l'apparence de vos menus. La liste de commandes est délimitée par des parenthèses () et les instructions sont séparées par le signe deux points ":". Par exemple:

```
Menu$(1)="(Locate 10,10 : Ink 1,1) Bonjour"
```

Chaque instruction se compose de deux caractères pouvant être affichés en minuscule ou en majuscule. Les autres caractères seront rejetés. La plupart des commandes vous demandent également d'entrer un ou deux nombres. Ces nombres **ne doivent jamais** former des expressions puisque celles-ci ne sont pas calculées. Les commandes sont listées ci- dessous.

Note: Dans la *syntaxe*, les deux caractères importants composant la commande sont en majuscule et en gras.

BOB *(Tracer un bob)*

BOb n

Cette commande trace un bob numéro *n* à la position courante du curseur. Elle ne prend pas en compte le point chaud du bob. Toutes les coordonnées sont calculées en prenant l'angle supérieur gauche comme base. Notez également que la couleur zéro est d'habitude considérée comme transparente. Vous pouvez la modifier au moyen de la commande NOMASK depuis l'AMOS Basic. Par exemple:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Menu$(1)="(Bob 1) 1":Menu$(1,1)="(Bob 2) 2":Menu$(1,2)="(Bob3) 3"
Menu on : Wait Key
```

ICON *(Tracer une icône)*

ICon n

Cette commande trace une icône numéro *n* à la position courante du curseur. Notez qu'à la différence des bobs, la couleur zéro **n'** est **pas** normalement transparente. Référez-vous à la commande MAKE ICON MASK en Basic pour plus de précisions.

LOCATE *(Déplacer le curseur graphique)*

LOcate x,y

Cette commande déplace le curseur graphique aux coordonnées *x,y* calculées en prenant l'angle supérieur gauche de la ligne du menu **comme base**. Notez qu'après la validation d'une instruction, le curseur graphique sera toujours positionné en bas et à droite de l'objet que vous venez de dessiner. Ces coordonnées seront également utilisées pour déterminer l'emplacement d'autres articles dans votre menu de la manière suivante:

```
Menu$(1)="Exemple ":Menu$(1,1)="Localiser (Lo 50,50) en action "
Menu$(1,2)="Devinez mes coordonnées"
Menu on:wait key
```

INK *(Définir la couleur de l'encre et du papier)*

INk n,valeur

La commande INK affecte les index de couleurs pour les couleurs STYLO, PAPIER et CONTOUR. Voici une liste des diverses possibilités:

n	Effet
1	Définir la couleur STYLO du texte
2	Définir la couleur PAPIER
3	Définir la couleur CONTOUR

SFONT *(Définir la police de caractères)*

SFont n

La commande SFont initialise la police courante à la police **graphique** numéro n. Celle-ci sera utilisée dans tous les articles des menus suivants. Notez que vous **devez** appeler GET FONTS avant d'exécuter cette instruction, sinon elle pourra seulement utiliser les deux polices de la mémoire fixe. Voir le fichier **EXEMPLE 16.5.**

SSTYLE *(Définir le style de police)*

SStyle n

Cette commande positionne le style de la police courante sur n qui est une configuration binaire dont la syntaxe suivante est la suivante:

Bit	Effet
0	Force ce bit à un pour souligner vos caractères
1	Sélectionne les caractères gras
2	Active le mode italique

LINE *(Tracer une ligne)*

LIne x,y

La commande LINE trace une ligne depuis la position courante du curseur aux coordonnées graphiques *x,y*. Voir **le fichier EXEMPLE 16.6.**

SLINE *(Définir le motif de ligne)*

SLine p

Cette commande force le style de ligne à la configuration binaire contenue dans *p* et ce style de ligne sera alors utilisé lors de l'exécution ultérieure des commandes LINE. Puisque les paramètres ne doivent pas être représentés par des expressions, cette configuration devrait être convertie d'emblée en notation décimale. Vous trouverez une simple démonstration des styles de ligne possibles dans **le fichier EXEMPLE 16.7.**

BAR *(Tracer une barre)*

BAr x,y

Cette commande trace une barre rectangulaire depuis les coordonnées courantes du curseur jusqu'à *x,y*. Référez-vous **au fichier EXEMPLE 16.8** pour une démonstration.

PATTERN *(Tracer un motif)*

PAttern n

Cette commande change le motif de remplissage utilisé par la commande Bar par le style *n*. Chargez le fichier **EXEMPLE 11.8** du dossier du MANUEL d'AMOS pour une démonstration.

OUTLINE *(Trace le contour d'une barre)*

OUline flag

Cette commande trace le contour de toutes les barres dans la ligne de contour courante (encre 3). Une valeur de un active le contour et une valeur de zéro le désactive.

ELLIPSE *(Tracer une élipse)*

ELlipse r1,r2

Cette commande trace une forme ovale de rayons r1 et r2 aux coordonnées courantes du curseur. Pour tracer un cercle, il vous suffit de donner une même valeur à r1 et r2. Voir **le fichier EXEMPLE 16.9.**

PROC *(Appeler une procédure0*

PRoc NAME

L'intruction PROC vous permet d'appeler n'importe quelle procédure d'AMOS Basic directement depuis la ligne d'un menu. La procédure qui est appelée **ne** doit **pas** comprendre de paramètres, sinon une erreur de syntaxe sera produite.

Cette commande vous permet d'adapter le menu à vos besoins précis sans avoir à vous limiter aux commandes de menu disponibles. Afin d'exploiter ces fonctions, il vous faudra assimiler un peu de théorie.

Au début de votre procédure, les valeurs suivantes sont contenues dans les registres du 68000.

DREG(0) **X-Coord**

Ce registre contient l'abscisse graphique X de l'angle supérieur gauche de l'article du menu courant. Ne tracez pas vos graphiques sur la portion de l'écran se trouvant à la gauche de ce point parce que cela gênera le processus de retracé du menu et peut amener des effets indésirables.

DREG(1) **Y-Coord**

Ce registre contient l'ordonnée graphique Y de l'article de votre menu. Comme avec la coordonnée X, vous devez toujours limiter vos opérations de tracé à la région se trouvant dessous ce point pour éviter d'éventuelles erreurs.

DREG(2) **Etat des opérations de tracé**

Ce registre contient l'état courant des opérations du menu. S'il contient une valeur de 0 (faux), l'article du menu est en train d'être tracé. Dans ce cas, il vous faudra affecter DREG(0) et DREG(1) des coordonnées de l'angle inférieur droit de la zone du menu et sortir immédiatement de la procédure.

Si DREG(0) est affectée d'une valeur de -1 (vrai), vous êtes libre d'effectuer les opérations graphiques utilisées par votre procédure. Vous devez ensuite renvoyer les coordonnées d'angle inférieur droit de votre article dans les registres DREG(0) ET DREG(1) comme précédemment.

DREG(3) **Etat de l'article du menu**

Ce registre reçoit une valeur de -1 si le menu est mis en surbrillance et si la première chaîne du menu est affichée, sinon il contiendra la valeur 0.

DREG(4)

Ce registre est positionné sur VRAI si la branche du menu est sélectionnée initialement.

AREG(1) **Adresse de la zone réservée**

C'est l'adresse de la zone créée avec RESERVE. Elles est utilisée pour permettre à plusieurs procédures de communiquer entre elles. Voir RESERVE pour plus de précisions. La syntaxe générale de la procédure d'un menu est la suivante:

```
Procedure ITEM
   If DREG(2)
        X=DREG():Y=DREG(1)
        ...Tracer l'article...
   Endif
   DREG(0)=BX:Rem Abscisse de l'angle inférieur droit de l'article du menu
   DREG(1)=BY:Rem Ordonnée de l'angle inférieur de l'article
Endproc
```

Les dimensions de l'article du menu telles qu'elles sont affichées à l'écran sont déterminées en utilisant les coordonnées BX et BY. Celles-ci **doivent** être chargées dans les registres D0 et D1 avant de quitter la procédure puisqu'elles sont nécessaires à la création de la dernière barre du menu.

Lorsque vous vous trouvez dans votre procédure, vous pouvez effectuer la plupart des instructions AMOS ainsi que les autres procédures. Mais certaines instructions sont absolument interdites! Si vous utilisez ces commandes, vous n'obtiendrez pas de message d'erreur mais votre AMIGA peut subitement se bloquer.

- Ne changez **jamais** l'écran courant à l'intérieur d'un menu. Il y a de grandes chances pour que cela plante votre Amiga!
- Ne définissez pas ou ne redéfinissez pas une zone d'écran.
- Evitez l'utilisation d'instructions telles que WAIT, WAIT KEY ou INPUT, ou INKEY$ qui suspendent l'exécution de votre programme.
- Les opérations sur disquette sont absolument interdites!
- L'interception des erreurs sera ignorée. Si une erreur se produit, le menu se fermera et votre programme retournera à l'éditeur.

Si vous l'utilisez avec précaution, la commande PROC peut produire des effets époustouflants. Pour en obtenir une démonstration, chargez le fichier **EXEMPLE 16.10** du dossier du MANUEL d'AMOS. Voir MENU CALLED.

RESERVE *(Réserver une zone locale de données pour une procédure)*

REserve n

Cette commande réserve *n* octets de mémoire pour cet article du menu. Vous pouvez accéder à cette zone depuis la procédure de votre menu en utilisant l'adresse contenue dans AREG(1). La zone de données que vous avez créée est commune à toutes les chaînes de l'objet courant du menu. Elle peut être utilisée pour échanger les paramètres entre les diverses procédures appelées par l'article d'un menu.

MENU CALLED (Retracer l'article d'un menu itérativement)

MENU CALLED(,,)

La commande MENU CALLED retrace automatiquement le menu sélectionné 50 fois par seconde à son affichage à l'écran. Elle est habituellement utilisée avec une procédure de menu pour produire des articles animés qui changeront devant vos yeux.

Pour vous servir de cette fonction, il vous faut d'abord définir une procédure de menu, en utilisant les principes exposés ci-dessus. Suppléez alors cette procédure d'un appel sous la forme appropriée de chaînes de titre au moyen de la commande intégrée PRoc. Finalement, activez le processus d'actualisation par un appel à MENU CALL. Lorsque vous affichez l'article choisi, votre procédure sera sollicitée régulièrement par le système de menu.

Puisque votre procédure est appelée cinquante fois par seconde, elle doit revenir sur le menu aussi rapidement que possible. Cela laisse assez de temps au reste du menu pour être parfaitement actualisé.

Notez également que votre procédure intégrée peut animer votre article au moyen de bobs ou de sprites. Cependant, puisque les articles du menu **ne** sont **pas** double tamponnés, il se peut que vos bobs scintillent légèrement à l'écran. Ainsi, il est préférable d'utiliser des sprites calculés à cette fin. Une autre approche consisterait à effectuer vos tracés à l'aide des commandes graphiques classiques d'AMOS. Vous trouverez une illustration dans **le fichier EXEMPLE 16.11** du dossier MANUEL.

MENU INIT *(Désactiver le retracé automatique)*

MENU INIT(,,)

Cette commande désactive le système d'actualisation automatique qui a été lancé au moyen de la commande MENU CALLED. A partir de maintenant, chaque article du menu est seulement retracé une seule fois sur l'appel du menu à l'écran.

Menu INIT(1,1)

D'autres styles de menu

Normalement les titres d'un menu sont affichés horizontalement et les options sont disposées en dessous dans une barre de menu verticale. Si vous voulez créer quelque chose d'un peu inhabituel, vous pouvez changer la structure de chaque niveau de votre menu en utilisant trois instructions:

MENU LINE *(Afficher un menu en disposant ses articles sur une ligne horizontale)*

MENU LINE niveau
MENU LINE (,,)

La commande MENU LINE affiche les options du menu au niveau demandé sur des lignes horizontales. Cette ligne de menu démarre à l'angle gauche du premier titre et s'étend jusqu'à l'angle inférieur droit du dernier titre.
MENU LINE niveau

Cette commande définit le style de menu de tout un niveau de votre menu. Elle doit être seulement appelée au cours des définitions de votre menu.

MENU LINE (,,)

Normalement, vous utilisez seulement la version *niveau* de cette commande. La définition d'articles sur Line et Bar peut donner des résultats bizarres mais cela peut être utile à quelque chose!

MENU TLINE *(Afficher un menu sur une seule ligne)*

MENU TLINE niveau
MENU TLINE(,,)

La commande MENU TLINE affiche une portion du menu sur une *seule ligne* de l'extrême gauche de l'écran à son extrême droite. Toute la ligne sera tracée même si le premier article se trouve au milieu de l'écran.

niveau est un numéro compris entre 1 et 8 spécifiant la partie du menu qui sera affectée. C'est la forme classique de cette instruction et cette dernière doit être appelée au cours des définitions de votre menu sinon elle n'aura aucun effet.

Vous pouvez également changer l'apparence d'un menu en utilisant une deuxième variante de cette commande. Par exemple:

```
Menu Line(1,1) : Rem Affiche le menu 1,1 sur une ligne
```

MENU BAR (Afficher une portion du menu sous la forme d'une barre)

MENU BAR niveau
MENU BAR(,,)

Cette commande affiche les articles sélectionnés au menu sous la forme d'une barre verticale. La définition de la largeur de cette barre est fonction des dimensions du plus grand article de votre menu.

niveau est un numéro indiquant la partie de la définition du menu courant qui sera affectée. Par défaut, cette option est utilisée pour les *niveaux de 2 à 8* de votre menu. Notez que la forme de cette instruction MENU BAR peut seulement être utilisée lors de l'initialisation de vos programmes. Si vous l'appelez après avoir activé un menu, elle

n'aura plus aucun effet.

(,,) représente la liste de paramètres vous permettant de changer le style de vos menus après leur installation. Voici un exemple de Menu Bar et Menu Tline:

```
FLAG=0
SET_MEN
Do
   If Choice and Choice(1)=2 and Choice(2)=1 Then ALTER
Loop
Procedure SET_MENU
   Menu$(1)="Démo barre " : Menu$(2)=" Sélectionnez Dessous "
   Menu$(1,1)=" Je ne fais rien! "
   Menu$(2,1)=" Oui, appuyez sur moi! "
   Menu On
End Proc
Procedure ALTER
   Shared FLAG
   Menu Del
   If FLAG=0 Then Menu Bar 1 : FLAG=1 Else Menu Tline 1 : FLAG=0
   SET_MENU
End Proc
```

MENU INACTIVE *(Désactiver les articles au menu)*

MENU INACTIVE niveau
MENU INACTIVE (,,)

Comme son nom le suggère, MENU INACTIVE désactive une série d'options de votre menu. Tout essai ultérieur de sélection de ces articles sera complètement rejeté. niveau vous permet de désactiver toute une section de menu et ses options au moyen de paramètres en (,,). Ceux-ci indiquent la position précise de votre article dans l'arborescence du menu courant.

Notez que les articles que vous avez désactivé au moyen de cette instruction seront immédiatement remplacés par la chaîne INACTIVE$ que vous avez spécifiée lors de la définition initiale de votre menu.

MENU ACTIVE *(Activer un article du menu)*

MENU ACTIVE niveau
MENU ACTIVE (,,)

La commande MENU ACTIVE renverse l'effet produit par la commande MENU INACTIVE. *niveau* choisit tout un niveau de menu à relancer. (,,) active un seul article de votre menu.

Après avoir appelé cette instruction, les options sélectionnées se réafficheront automatiquement en utilisant leurs chaînes de titre d'origine.

Les menus déplaçables

Vous pouvez afficher les menus AMOS à n'importe quel point de l'écran de votre Amiga et les déplacer directement par vous-même ou explicitement par votre programme.

MENU MOVABLE (Activer le déplacement automatique des menus)

MENU MOVABLE niveau
MENU MOVABLE(,,)

La commande MENU MOVABLE informe le système de menus que les articles du *niveau* peuvent être déplacés par l'utilisateur; c'est le choix effectué par le système.

La deuxième variante de cette commande vous permet de définir l'état de chaque article au menu. Les paramètres entre parenthèses indiquent la position de ces articles dans l'arborescence du menu.

Vour pouvez repositionner le menu en déplaçant la flèche de la souris sur le **premier** article du menu et en appuyant sur le bouton gauche de la souris. Une boîte rectangulaire apparaîtra alors, encadrant l'article sélectionné et vous pouvez la déplacer dans tout l'écran courant. Si vous relâchez le bouton gauche, le menu et ses articles se retraceront à cette nouvelle position.

Notez que cette commande ne vous permet pas de changer la disposition des articles se trouvant en dessous de ce niveau. Si vous voulez manipuler les options du menu, il vous faudra utiliser une commande MENU ITEM particulière. Référez-vous **au fichier EXEMPLE 16.12** pour étudier une démonstration de ce système.

MENU STATIC *(Immobiliser un menu)*

MENU STATIC niveau
MENU STATIC(,,)

Cette commande définit le menu à un *niveau* ne pouvant pas être déplacé par l'utilisateur. Le problème que l'on rencontre avec les menus déplaçables est que la quantité de mémoire qu'ils consomment varie au cours d'un programme. Si vos menus sont particulièrement grands ou si la capacité de la mémoire se rarifie, cela peut entraîner de véritables problèmes: une seule étourderie suspendra votre programme en faisant apparaître un message d'erreur *mémoire insuffisante*. Vous pouvez éviter cette difficulté en utilisant la commande MENU STATIC.

MENU ITEM MOVABLE *(Déplacer les options du menu)*

MENU ITEM MOVABLE niveau
MENU ITEM MOVABLE(,,)

Cette commande est similaire à MENU MOVABLE sauf qu'elle vous permet de redisposer les diverses options à un certain niveau. Ainsi, vous pouvez repositionner chacun des articles se trouvant dans une barre de menu.

Normalement, il est interdit de déplacer les articles en dehors de la barre courante du menu, mais cette interdiction peut être outrepassée au moyen de la commande MENU SEPARATE.

Pour pouvoir bouger les articles du menu, **toute** la barre du menu doit pouvoir être déplacée. Ainsi, si vous immobilisez le MENU au moyen de MENU STATIC, cette commande aura aucun effet. En outre, vous ne pouvez pas bouger le premier article de la barre du menu puisque cela déplacerait toute la ligne. Un autre effet indésirable est que le déplacement du **dernier** article du menu réduira définitivement la taille de la barre de votre menu. Il existe trois solutions possibles à ce problème:

• Délimitez toute la barre par une boîte rectangulaire de la manière suivante:

Menu$(1,1)=,,,"(Bar 40,100)(Loc 0,0)"

Si MENU$(1,1) est le premier article de votre barre courante

• Immobilisez le dernier article à l'aide de MENU ITEM STATIC.

MENU ITEM STATIC *(Article permanent du menu)*

MENU ITEM STATIC niveau
MENU ITEM STATIC(,,)

Cette commande bloque un ou plusieurs articles du menu et elle correspond également au positionnement implicite.

MENU SEPARATE *(Séparer une liste d'articles du menu)*

MENU SEPARATE niveau
MENU SEPARATE(,,)

La commande MENU SEPARATE ordonne à AMOS de séparer tous les articles du niveau courant. Chaque article de votre menu est considéré tout à fait indépendamment du précédent. Si vous n'avez pas défini une chaîne d'arrière plan, chaque article sera décalé de deux pixels par rapport à celui du dessus. Cela crée un joli effet de pallier qui peut être annulé en éditant le menu au moyen de l'accessoire du MENU.

Les paramètres optionnels de cette instruction vous permettent de diviser une barre de menu à n'importe quel point de la ligne. L'article séparé sera affecté par les commandes MENU MOVABLE et non par les instructions ITEM.

MENU LINKED *(Relier un groupe de menus)*

MENU LINKED niveau
MENU LINKED(,,)

Cette commande relie un ou plusieurs articles du menu. Elle est l'opposé de l'instruction

MENU SEPARATE.

=MENU X *(Renvoyer l'abscisse graphique de l'article d'un menu)*

x=MENU X(,,)

La fonction MENU X vous permet de retrouver la position de l'article d'un menu en fonction de la position de celle qui précéde. Vous pouvez utiliser ces informations pour mettre en place des menus puissants tels que celui que vous trouverez dans **le fichier EXEMPLE 16.13.** =MENU Y (Renvoyer l'ordonnée graphique de l'article d'un menu)

y=MENU Y(,,)

Cette fonction renvoie l'ordonnée Y de l'option d'un menu. Notez que toutes les coordonnées sont calculées en fonction des positions de l'article précédent. Ainsi, ce n'est **pas** une ordonnée d'écran normale.

Déplacer un menu dans un programme

MENU BASE *(Déplacer le point de départ d'un menu)*

MENU BASE x,y

Cette commande déplace le point de départ du premier niveau de vos menus vers les coordonnées absolues de l'écran *x,y*. Tous les articles inférieurs seront affichés à leur position courante par rapport au sommet de votre menu. Référez-vous **au fichier EXEMPLE 16.14** vous donnant une démonstration de la commande MENU BASE.

SET MENU *(Déplacer urt menu)*

SET MENU(,,) To x,y

Cette commande définit les coordonnées de l'angle supérieur gauche de l'article d'un menu. Ces coordonnées sont calculées **en rapport** au niveau précédent. Le point de départ du menu tout entier (coordonnées 0,0) peut être déterminé au moyen de la commande MENU BASE.

Tous les niveaux des sous-menus seront également déplacés par cette instruction. Leur position relative restera inchangée. Puisque x,y peuvent avoir une valeur négative, il est possible de disposer les articles dans une barre de menu sous la forme d'un panneau de commandes; voir le fichier **EXEMPLE 16.15.**

Afficher un menu à la position du curseur

MENU MOUSE *(Afficher le menu sous la souris)*

MENU MOUSE ON/OFF

Les fonctions de MENU MOUSE affichent automatiquement tous les menus depuis la position courante du curseur. Les coordonnées de la souris sont ajoutées à celles de MENU BASE pour obtenir la position finale. Il est donc possible de positionner le menu à une distance fixe de la flèche de la souris si besoin. Voir **le fichier EXEMPLE 16.16.**

17 Musique et effets sonores

Le système générateur de sons de l'Amiga est capable de produire des effets sonores stéréo qui vous auraient paru inouïs il y a quelques années. Les résultats sont impressionnants même avec votre télé, mais si vous branchez votre Amiga à une chaîne Hi-Fi, les sons restitués peuvent vraiment faire trembler les murs de la pièce dans laquelle vous vous trouvez!

Comme vous pouvez vous y attendre, nous avons fait du chemin depuis la modeste commande BIP. En fait, nous vous avons proposé tout ce dont vous avez besoin pour que vous puissiez intégrer des effets sonores hallucinants à vos jeux. L'exécution de toutes les commandes sonores d'AMOS se fait indépendamment de vos programmes Basic. Ainsi, vous pouver passer et repasser vos bandes d'enregistrement sans affecter du moins la qualité du déroulement du jeu.

Vous pouvez créer des échantillons en utilisant les cartouches d'échantillonnage qui vous sont proposées et les rejouer au moyen d'une simple instruction SAMPLAY. Chaque échantillon peut être joué à différentes allures et réitéré. Il est même possible d'avoir un échantillon comprenant une seule note de musique.

Vous pouvez convertir la musique d'un progiciel tel que SONIX, SOUNDTRACKER ou GMC. Le système générateur de musique d'AMOS est intelligent et s'arrêtera automatiquement si un son est joué par le canal courant, vous permettant ainsi de combiner facilement des morceaux et la musique dans le même canal sonore sans craindre d'effets perturbateurs indésirables.

Chaque chanson peut être accompagnée de 256 instruments; la seule limite imposée au nombre de chansons est la capacité de la mémoire. Afin de maintenir les temps de gestion de la mémoire au strict minimum, toutes les mélodies sont composées à partir d'un certain nombre de motifs particuliers. Vous pouvez intégrer un motif à votre morceau en utilisant seulement deux ou trois octets. En définissant quelques motifs clé, vous pouvez donc créer des douzaines de mélodies sans vous trouver à court de mémoire.

L'atout du système générateur de musique d'AMOS est son extensibilité. Vous trouverez l'intégralité du code source sur la disquette de données. Ainsi, les développements à venir dans le monde musical de l'Amiga ne vous laisseront pas en plan.

Les effets sonores simples

Nous allons commencer par vous présenter les effets sonores intégrés prédéfinis dans l'AMOS Basic. Ceux-ci sont l'équivalent AMOS de la commande BIP de l'Amiga Basic.

BOOM *(Produire un bruit simulant celui d'une explosion)*

BOOM

Kapout! Vous êtes mort! Utilisez BOOM pour intégrer cet effet sonore dans vos jeux. Ce

type de "bruit blanc" a été extrêmement difficile à créer sur l'Amiga mais AMOS utilise un système d'interruption ingénieux pour produire un effet réel d'explosion. Exemple:

```
Boom : Print "Vous êtes MORT!"
```

SHOOT *(Produire un bruit simulant celui d'un coup de feu)*

SHOOT

La commande SHOOT produit un simple effet de coup de feu. Comme BOOM, SHOOT ne suspend en aucune façon votre programme. Par conséquent, si vous tirez plusieurs coups de feu successifs, vous pourriez ajouter une petite temporisation au moyen de la commande WAIT.

```
Shoot : Wait 6 : Shoot : Print "Vous êtes MORT!"
```

BELL *(Simple son de cloche)*

BELL [f]

La commande BELL produit un son pur de fréquence *f*. *f* définit la hauteur d'une note, comprise entre 1 (très grave) et 96 (très aigu). Exemple:

```
Bell : Wait 40 : Rem Attendre l'exécution de la sonnerie
For F=1 To 96
   Bell F:Wait F/10+1 : Rem Varier la temporisation et la fréquence
Next F
```

Les canaux sonores

La machine de l'Amiga peut facilement émettre en même temps quatre sons différents. Cela vous permet d'ajouter de jolies harmoniques à vos effets sonores.

Chaque son peut être émis sur une des quatre *voix* numérotées de 0 à 3. Vous pouvez les imaginer comme étant des instruments indépendants qui jouent leurs séquences de notes, leurs morceaux ou leurs échantillons. Les quatre voix sont combinées dans le système pour produire le son final qui sera restitué par votre enceinte.

Les instructions sonores d'AMOS vous permettent de créer les sons que vous voulez en utilisant n'importe quelle disposition de voix. Toutes les commandes sonores d'AMOS peuvent générer les sons de votre choix selon la disposition de voix que vous souhaitez. Chaque voix est affectée à un bit particulier dans un paramètre VOICE de la manière suivante:

```
Bit 0-> Voix 0
Bit 1-> Voix 1
Bit 2-> Voix 2
Bit 3-> Voix 3
```

Pour activer les voix souhaitées, forcez les bits appropriés à 1. Voici une liste des valeurs courantes pour vous faciliter un peu la vie.

Valeur	Voix utilisées	Effet
15	0,1,2,3	Utilise les quatre voix.
9	0,3	Ces voix sont combinées et l'effet généré est restitué par l'enceinte de gauche.
8	3	
6	2,4	Emis par l'enceinte de DROITE.
4	2	
2	1	
1	0	

Pour vous rendre compte de la qualité de ces effets sonores, il vous faudra sans doute brancher votre Amiga à une chaîne hi-fi. La plupart des télévisions ne sont pas capables de reproduire toute la gamme de sons pouvant être générée par le super hardware de l'Amiga.

VOLUME *(Moduler le volume sonore)*

VOLUME [v,] intensité

La commande VOLUME module le volume des sons devant être émis par un ou plusieurs canaux sonores.

Intensité se rapporte à la force de ce son. L'intensité est normalement comprise entre 0 (silencieux) à 63 (très fort). Par défaut, le volume est de même intensité pour les quatre voix possibles. Le nouveau volume sera utilisé pour tous les effets sonores ultérieurs, y compris la musique.

Le paramètre *v* vous permet de changer le volume de chaque voix. *v* indique maintenant la combinaison de voix à moduler. Cette deuxième option est seulement utilisée par les effets sonores. Elle n'a pas d'effets sur les morceaux de musique que vous jouez. Les voix sont sélectionnées au moyen du mode de syntaxe binaire, dont chaque bit représente l'état d'un seul canal sonore. Si le bit est forcé à 1, alors le volume de cette voix sera modulé, sinon il ne sera pas du tout affecté. Exemples:

```
Volume %0001,63 : Boom : Wait 100 : Rem Régler le Canal 1 sur le volume 63
Volume %1110,10 : Boom : Wait 50 : Rem Les Canaux 2,3,4 ont un volume de 10
Play 40,0 : Wait 30
Volume 50 : Play 40,0
```

Son échantillonné

Si vous devez produire tous les effets sonores dont vous avez besoin, directement à l'intérieur de votre ordinateur, vous seriez mis en face d'une tâche impossible. En pratique, il est souvent plus facile de prendre un son d'une source externe, telle qu'un magnétophone et de le convertir en une liste de nombres pouvant être conservée dans la mémoire de votre ordinateur.

Chaque nombre représente le volume d'un échantillon particulier du son. En rejouant rapidement ces valeurs par l'intermédiaire des circuits générateurs de sons, vous pouvez reproduire une réelle impression du son d'origine. Cette technique est à la base de la production d'effets sonores échantillonnés, que l'on rencontre dans la plupart des logiciels de jeu.

Si vous voulez créer vos propres échantillons, il vous faudra ajouter un autre élément matériel à votre ordinateur, une CARTOUCHE ECHANTILLONNEUR. Bien que ces cartouches soient très amusantes, elles ne sont certainement pas importantes. AMOS Basic est parfaitement capable de jouer n'importe quel échantillon sonore, sans avoir besoin de ces suppléments onéreux.

Il existe actuellement des centaines d'effets sonores en vente dans le domaine publique, couvrant la plupart des effets dont vous avez besoin dans vos jeux. Nous avons même prévu un choix d'échantillons utiles sur la disquette de données AMOS pour vos expérimentations.

SAM PLAY *(Jouer un échantillon sonore de la banque d'échantillons d'AMOS)*

SAM PLAY s
SAM PLAY v,s
SAM PLAY v,s,f

L'instruction SAM PLAY vous permet d'émettre un son échantillonné directement par vos enceintes. Tous les échantillons sont normalement conservés dans la banque mémoire numéro cinq, mais vous êtes libre de les changer de banque en utilisant la commande SAM BANK.

s est le numéro de l'échantillon qui sera joué. Il n'existe aucune limite imposée au nombre d'échantillons que vous pouvez mémoriser dans une banque, autre que la capacité de la mémoire dont vous disposez. Si vous voulez utiliser vos échantillons avec cette instruction, il vous faudra les intégrer dans une banque mémoire d'AMOS. Vous trouverez toutes les précisions à ce sujet en fin de chapitre.

v est un mode point contenant une liste de voix utilisées par votre échantillon. Comme d'habitude, il y a un bit pour chaque voix possible. Pour jouer vos échantillons sur la voix demandée, il vous suffit de forcer le bit correspond à 1. Référez-vous à l'explication précédente des canaux sonores pour plus de détails.

f représente la vitesse de lecture de votre échantillon, mesurée en *hertz*. Elle spécifie le nombre d'échantillons joués par seconde. Les vitesses typiques d'échantillons vont de 4000 pour les bruits tels que les explosions à 10000 pour les effets vocaux reconnaissables. En modifiant la vitesse de lecture, vous pouvez facilement régler la hauteur du son émis sur une plage importante. Ainsi, un seul échantillon peut être utilisé pour produire des douzaines de sons différents. Exemples:

```
Rem Charger la banque d'échantillons avec quelques échantillons de la disquette de
données d'AMOS
Load "AMOS_DATA:SAMPLES/SAMPLE_DEMO.ABK"
For S=1 to 11
   Locate 0,0: ? "Jouer un échantillon ";S
   Sam Play S
```

```
Locate 0,24 : Centre "<Frappez une touche pour continuer>" : Wait Key : Cline
Next S
Wait Key
Sam Play 1,11 : Wait 5 : Sam Play 2,11 : Rem simple effet d'écho
Wait key
Sam Play 1,1,2000 : Rem Son grave
Wait Key
Sam Play 1,1,15000 : Rem Son aigu
```

Vous trouverez une autre démonstration de cette commande dans **le fichier EXEMPLE 17.1** du dossier du MANUEL.

SAM BANK *(Changer la banque courant d'échantillons)*

SAM BANK n

La commande SAM BANK affecte une nouvelle banque mémoire destinée à recevoir vos échantillons. Toutes les instructions ultérieures SAM PLAY saisiront leurs sons directement de cette banque.

Il est possible d'exploiter cette fonction pour que la mémoire contienne plusieurs jeux complets d'échantillons l'un à côté de l'autre. Vous pouvez parcourir ces échantillons à votre gré en appelant simplement la commande SAM BANK.

SAM RAW *(Jouer un échantillon depuis la mémoire)*

SAM RAW voix,adresse,longueur,fréquence

La commande SAM RAW joue un échantillon de la mémoire de l'Amiga.
voix est une configuration binaire de syntaxe classique qui spécifie la liste de voix utilisées par votre échantillon. Chaque bit de la configuration sélecte un seul canal (voir les canals sonores).

adresse détient l'adresse de votre échantillon. Normalement, celle-ci se rapporte au corps d'une banque mémoire d'AMOS existant. longueur comprend la *longueur* de l'échantillon que vous souhaitez jouer. *fréquence* indique la vitesse de lecture de l'échantillon sélectionnée (en échantillons par seconde ou Hz). Cette vitesse peut être différente de celle à laquelle l'échantillon fut initialement enregistré.

SAM RAW vous permet de restituer les échantillons classiques de l'Amiga directement par vos enceintes, sans avoir le besoin de créer une banque mémoire spéciale (voir *Créer une banque échantillons*). Il est maintenant de votre ressort de gérer vos échantillons dans la mémoire et de taper les paramètres des échantillons. SAM RAW est bien utile pour parcourir les fichiers de votre collection de disquettes. Utilisez BLOAD pour saisir un fichier d'une banque puis SAM RAW pour émettre ces données. Avec un peu de chance, vous devriez obtenir des sons intéressants. Exemples:

```
Reserve as work 10,55000
```

```
Bload "Samples/Samples.abk",start(10)
Sam Raw 15,start(10),length(10),10000
```

SAM LOOP *(Réproduire un échantillon)*

SAM LOOP ON/OFF

L'instruction SAM LOOP informe AMOS Basic que tous les échantillons ultérieurs doivent être bouclés à l'infini. Exemples:

```
Rem Charger la banque échantillons avec quelques échantillons de la disquette AMOS DATA
Load "AMOS_DATA:SAMPLES/SAMPLES.Abk"
Sam Loop On
For S=1 to 11
   Locate 0,0 : Print "Jouer un échantillon";S
   Sam Play S
   Locate 0,24 : Centre"<Frappez une touche pour continuer>" : Wait Key : Cline
Next S
Sam Loop Off
```

Vous pouvez désactiver cette itération par un simple appel à la commande SAM LOOP OFF.

Créer une banque échantillons

Si vous souhaitez jouer vos propres échantillons en utilisant SAM PLAY, il vous faudra d'abord les charger dans une banque mémoire. Pour cela, vous devez sélectionner le programme SAMMAKER se trouvant sur la disquette de données AMOS.

A la mise en route, SAMMAKER vous présente un classique sélecteur de fichier AMOS. Entrez le nom du fichier du premier échantillon à conserver dans votre nouvelle banque et appuyez sur RETOUR. Si AMOS ne peut pas trouver la vitesse d'échantillonnage, il vous sera demandé de l'entrer directement. Si vous faites une erreur à cette étape, cela n'a pas vraiment d'importance puisque vous pouvez rejouer vos échantillons sans crainte, à la vitesse que vous souhaitez.

Après une courte attente, vous serez invité à installer l'échantillon suivant dans la banque. Lorsque vous avez terminé l'installation de tous vos échantillons, tapez SAUVEGARDER au sélecteur de fichier pour sauvegarder vos échantillons sur la disquette. Il vous sera automatiquement demandé d'entrer le nom du fichier de destination de votre nouvelle banque, ce que vous ferez au moyen de la commande CHARGER comme suit:

```
Load "sample.abk"
Load "sample.abk",6 : Rem charger les échantillons dans la banque 6
```

Musique

Le générateur de musique d'AMOS vous permet de créer une musique de fond accompagnant vos jeux. Elle peut être composée depuis diverses sources, comme

GMC, SOUNDTRACKER ou SONIX.

Pour convertir ces morceaux de musique en un format spécial AMOS, il vous faudra utiliser un des programmes de conversion se trouvant sur la disquette de données AMOS. Vous devez sauvegarder la musique GMC au moyen de l'icône SAUVEGARDER LES DONNEES, puisque ceci copie à la fois la musique et les définitions d'instrument sur un seul grand fichier de données.

MUSIC *(Jouer un morceau de musique)*

MUSIC n

La commande AMOS MUSIC permet de jouer un morceau de musique de la banque musique (trois). Cette musique sera jouée indépendamment de votre programme Basic, sans l'affecter pour le moins .

Il est normalement possible d'enregistrer plusieurs agencements complets dans la même banque. Chaque disposition est affectée de son propre numéro de musique. La seule exception à cette règle est que la musique composée par GMC vous permet seulement de placer une seule chanson à la fois dans une banque. Exemple:

```
Rem Charger un morceau de musique depuis la disquette de données AMOS
Load "MUSIC/music.Abk"
Music 1
```

Le générateur de musique AMOS est intelligent: il suspend automatiquement votre musique pendant l'émission d'effets sonores sur le canal courant et votre mélodie reprendra là elle a été interrompue.

Trois airs peuvent être joués en même temps. Chaque nouvelle commande musicale arrête la chanson en cours et la met sur une *pile*. Lorsque la chanson est terminée, la musique précédente reprendra là où elle a été interrompue.

MUSIC STOP *(Arrêter un seul morceau de musique)*

MUSIC STOP

Cette commande interrompt le morceau de musique en cours. Si un autre morceau a été activé, il reprendra immédiatement.

MUSIC OFF *(Désactiver toute musique)*

MUSIC OFF

Cette commande désactive complètement votre musique. Pour la réactiver, il vous faut réexécuter votre série d'instructions MUSIC depuis le début.

TEMPO *(Changer l'allure d'un échantillon musical)*

TEMPO s

Cette commande modifie l'allure de l'air en cours, joué à l'aide de la commande MUSIC. s représente la nouvelle allure et peut être comprise entre 1 (très lente) et 100 (très rapide). Cependant, les instruments ne sont pas tous capables de jouer à l'allure maximale suggérée. La limite pratique est proche de 50.

Pour une démonstration, placez la disquette de données AMOS dans le lecteur de disquette courant et tapez:

```
Load "AMOS_DATA:MUSIC/music.Abk" : Rem Chargez la musique
Music 1 : Rem Jouez la musique 1
Tempo 35 : Rem Régler l'allure de la musique sur très rapide
Tempo 5 : Rem Régler l'allure de la musique sur très lent
```

Notez que la musique composée à l'aide de GMC contient souvent des étiquettes fixant le tempo directement dans la disposition. Ces étiquettes écraseront les définitions du tempo dans l'AMOS Basic. Nous vous conseillons donc pas de les utiliser dans votre propre musique.

MVOLUME *(Régler le volume d'un morceau de musique)*

MVOLUME n

Cette commande change le volume de tout le morceau de musique sur l'intensité n. n est compris entre 0 (silencieux) et 63 (très fort).

VOICE *(Activer une ou plusieurs voix d'un morceau de musique)*

VOICE masque

Cette commande active une ou plusieurs voix du morceau de musique. Habituellement, chaque voix renferme sa propre mélodie qui sera restitués par vos enceintes pour produire votre musique finale.

masque est un masque binaire de format AMOS classique qui spécifie les voix que vous souhaitez jouer. Chaque bit représente l'état d'une voix dans la musique. S'il est forcé à 1, la voix sera jouée, sinon elle ne sera pas utilisée. Exemples:

```
Load"MUSIC/music.abk" : Rem Charger la musique
Music 1 : Rem Jouer la musique 1
For V=0 To 15
Locate 0,0 : Print "Voix";V
Voice V
Wait 100
Next V
Direct
Voice %0001 : Rem Activer la voix 0
```

```
Voice %0010 : Rem Voix 1
Voice %1001 : Rem Voix 3 et 0
Voice %1111 : Rem Voix 4
```

=VUMETER *(Mesure du volume)*

s=VUMETER(v)

La fonction VUMETER teste la voix *v* et renvoie le volume de la note courante qui est jouée par votre morceau de musique. *s* représente une valeur d'intensité comprise entre 0 et 63. *v* est le numéro d'une seule voix à vérifier (de 0 à 3).

A l'aide de cette fonction, vous pouvez faire dancer vos sprites sur un morceau de musique! Chargez **le fichier EXEMPLE 17.2** pour une démonstration.

Remarquez qu'il existe seulement une version AMAL de cette instruction vous permettant de créer des VU mètres en temps réel en interruption. Reportez-vous au paragraphe sur la commande VU pour avoir plus de précisions.

Jouer une note

PLAY *(Jouer une note)*

PLAY [voix,] hauteur,attente

Cette commande joue une seule note qui est émise par l'enceinte de votre télé ou de votre chaîne Hi-Fi. *Hauteur* fixe la hauteur de ce son, comprise entre 0 (bas) et 96 (haut). Au lieu d'être seulement un nombre arbitraire, chaque hauteur est associée à une des notes (Do, Ré, Mi, Fa, Sol, La, Si) comme vous pouvez le voir dans le tableau suivant:

	Note											
	Do	**Do#**	**Ré**	**Ré#**	**Mi**	**Fa**	**Fa#**	**Sol**	**Sol#**	**La**	**La#**	**Si**
Octave												
0	1	2	3	4	5	6	7	8	9	10	11	12
1	13	14	15	16	17	18	19	20	21	22	23	24
2	25	26	27	28	29	30	31	32	33	34	35	36
3	37	38	39	40	41	42	43	44	45	46	47	48
4	49	50	51	52	53	54	55	56	57	58	59	60
5	61	62	63	64	65	66	67	68	69	70	71	72
6	73	74	75	76	77	78	79	80	81	82	83	84
7	85	86	87	88	89	90	91	92	93	94	95	96

Il devrait être évident que les notes montent par cycle de 12. Ce cycle est connu sous le nom d'octave.

Le paramètre optionnel *voix* vous permet de jouer vos notes sur n'importe quelle combinaison des quatres voix de l'Amiga. Comme de coutume, c'est un mode point dont la syntaxe est la suivante:

Bit 0-> Voix 0
Bit 1-> Voix 1
Bit 2-> Voix 2
Bit 3-> Voix 3

En forçant un bit à 1, vous pouvez jouer la note sur la voix correspondante.

attente fixe la durée de la pause entre la commande play et l'instruction Basic suivante. Ceci vous permet de jouer chaque note avant de procéder à la suivante.

Une temporisation de zéro lance une note et saute immédiatement à l'instruction Basic suivante. En jouant plusieurs notes l'une après l'autre, vous pouvez facilement produire de supers effets harmoniques. Exemples:

```
Play 1,40,0 : Play 2,50,0 : Rem Jouez sans temporisation
Wait Key
Play 1,40,15 : Play 2,50,15 : Rem Constatez l'effet de temporisation
Rem Jouez une suite de notes au hasard
Do
   T=Rnd(96) : V=Rnd(15) : Play V,T,3
Loop
```

La commande PLAY n'est pas simplement limitée à des bruits blancs. Il est également possible d'affecter des signaux complexes au générateur sonore à l'aide des puissantes commandes WAVE et NOISE.

Ondes et enveloppes

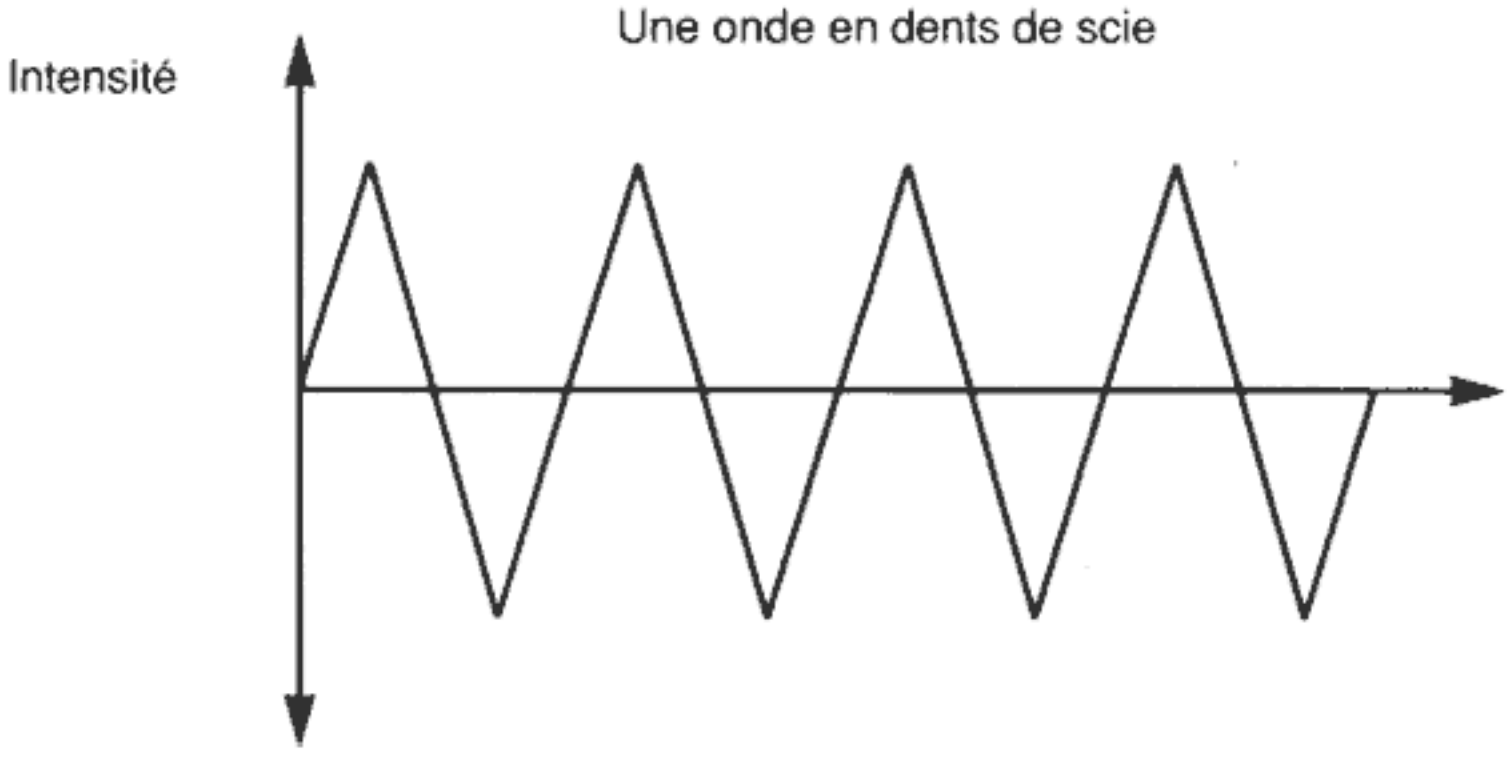
SET WAVE *(Définir la forme d'une onde)*

SET WAVE onde,forme$

L'instruction SET WAVE vous permet de définir vos propres instruments que vous manipulerez à l'aide de l'instruction AMOS Basic PLAY. Le son de votre instrument dépend de la forme de l'onde contenue dans la mémoire de l'Amiga. Ceci constitue un modèle qui est reproduit pour former votre note finale.

onde est le numéro de l'onde que vous souhaitez définir. Les numéros d'onde autorisés commencent à partir de 2 parce que les ondes zéro et 1 sont déjà installées. L'onde zéro représente une forme de forme irrégulière qui produit l'effet d'explosion. L'onde un est une onde sinusoïdale de forme régulière qui produit les sons purs utilisés par l'instruction normale PLAY.

Les formes des signaux sont définies en utilisant une liste de 256 nombres entrés en utilisant le paramètre SHAPE$. Voici un exemple d'un de ces signaux:



Chaque numéro représente l'intensité une section du signal. Il est l'équivalent à la hauteur d'un seul point au graphe ci-dessus.

Les valeurs possibles de l'intensité sont comprises entre -128 et 127. Puisque les chaînes d'AMOS sont seulement capables de contenir des nombres **positifs** (de 0 à 255), il vous faudra convertir vos valeurs négatives en une forme interne spéciale avant de les utiliser. Vous pouvez calculer la valeur choisie en ajoutant simplement 256 aux nombres négatifs de votre liste. Par exemple, la conversion de -50 est la suivante:

-50+256=206

Voici un programme démontrant comment une onde en dents de scie serait créée en AMOS Basic.

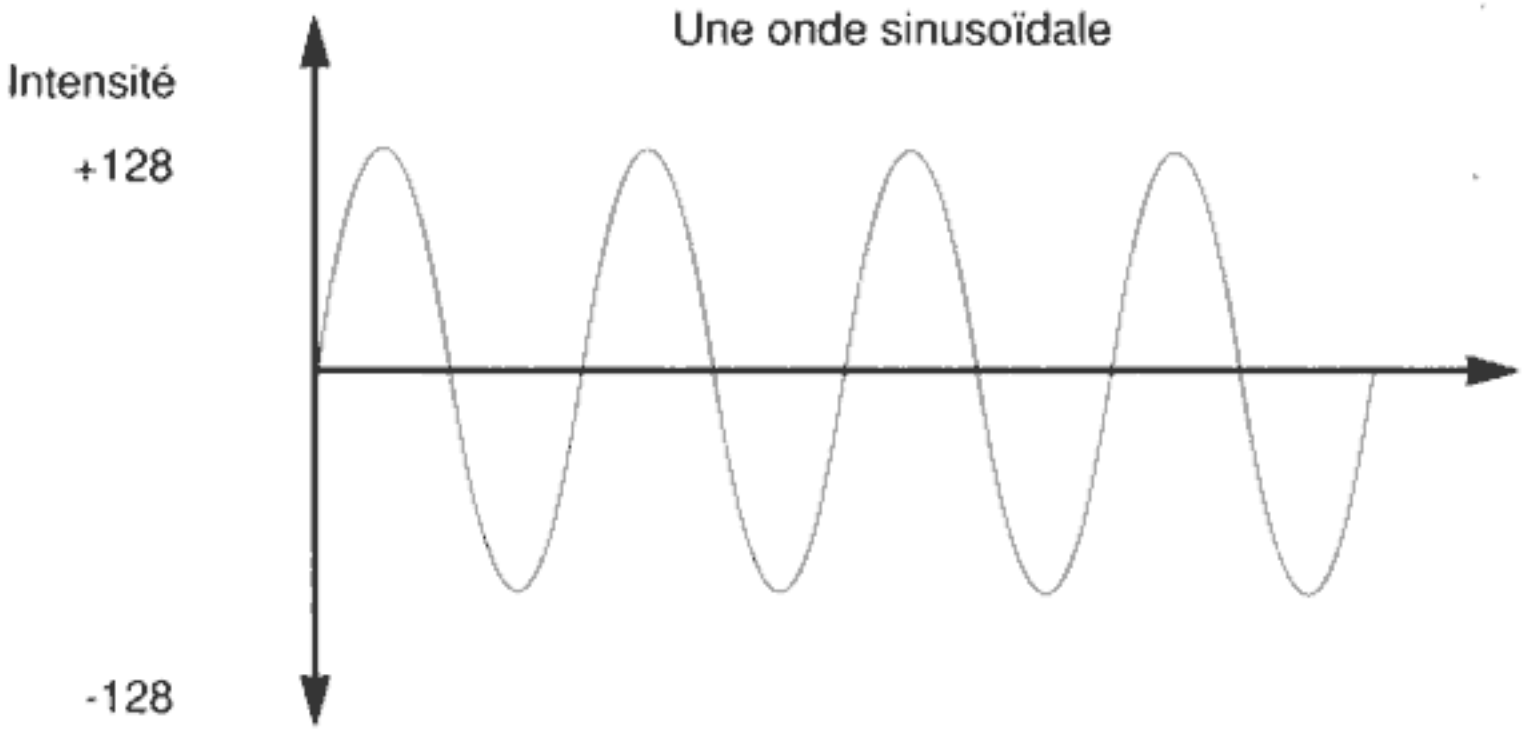
```
S$="" : Rem effacer la chaîne du signal
For I=-128 To 127
   X=I : If X<0 Then Add X,256
   S$=S$+Chr$(X)
Next I
Set Wave 2,S$
```

Avant d'émettre votre signal, vous devez indiquer les canaux qui lui seront affectés. Pour cela, utilisez la commande WAVE. Ajoutez la ligne suivante à la routine précédente:

```
Wave 2 To 15 : For S=10 to 60 : Play S, 10 : Next S
```

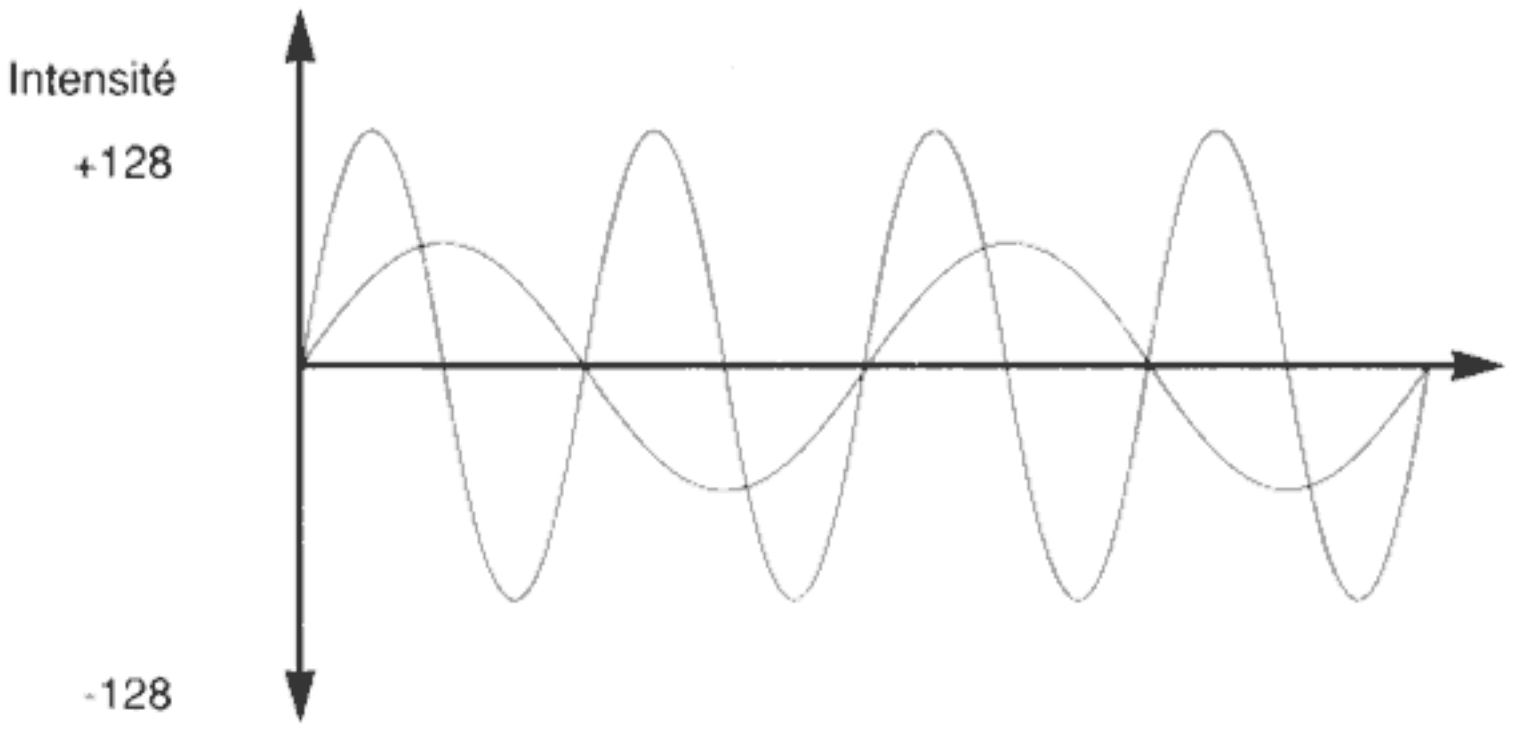
La meilleure façon de reproduire l'effet d'un véritable instrument est de combiner les ondes sinusoïdales. Un exemple de l'une de ces ondes sinusoïdales vous est donné dans le schéma ci-dessous.

Une onde sinusoïdale



La réunion de plusieurs de ces ondes aux tailles différentes et aux points d'émission distincts produit des ondes dont le schéma est le suivant:

Ondes complexes



Ceci produit des harmoniques régulières permettant de jouer vos notes. Voici un exemple:

```
SHAPE$="" : Degree
For S=0 To 255
   V=Int((Sin(S)/2+Sin(S*2+45)/4)*128)+127
   SHAPE$=SHAPE$+Chr$(V)
Next S
Set Wave 2,SHAPE$ : WAVE 2 TO 15
For N=10 To 60 : Play N,10 : Next N
```

WAVE *(Affecter une onde à un ou plusieurs canaux sonores)*

WAVE affecte un numéro d'onde w à un ou plusieurs canaux sonores. v représente un mode point de format classique. Si un bit de la configuration est forcé à 1, alors les voix correspondantes seront utilisées par PLAY, sinon elles ne seront pas du tout affectées. Par défaut, l'onde zéro est réservé au canal NOISE et l'onde un représente une onde sinusoïdale. Voici quelques exemples:

```
Wave 0 To %0001 : Rem Affecter une onde 0 à la voix zéro
Play 1,40,0
Wave 0 To %1100 : Rem Utiliser les voix 3,2 pour émettre un bruit
Play 20,10
Wave 1 To %1111 : Rem Jouer un son pur sur les quatre voix
Play 60,0
```

NOISE *(Affecter une onde sonore à un canal)*

NOISE To voix

Cette commande superpose un bruit blanc (onde 0) sur les voix sélectionnées. Elle forme la base d'une panoplie d'effets d'explosion et de percussion. Chargez **le fichier EXEMPLE 17.3** du dossier MANUEL pour une démonstration.

voix est une configuration binaire classique. Les quatre premiers bits représentent les quatre voix possibles sur l'Amiga, à partir de zéro. Si un bit est forcé à 1, alors le souffle sera émis sur ce canal, sinon l'instruction ne changera par la voix. NOISE est l'équivalent de la commande:

```
Wave 0 To voices
```

Exemples:

```
Noise To 15
Play 60,0
Play 30,0
```

DEL WAVE *(Supprimer une onde)*

DEL WAVE n

Cette commande supprime une onde préalablement définie par SET WAVE. *n* est le numéro de l'onde à partir de 2. Il est impossible de supprimer les ondes prédéfinies NOISE et SINE en utilisant cette instruction. Après avoir effacé une onde, toutes les voix seront redéfinies selon l'onde normale SINE (par défaut).

SAMPLE *(Affecter un échantillon à une onde)*

SAMPLE n To voix

Cette commande est la plus puissante de toutes les commandes d'ondes. Elle affecte un échantillon conservé dans la banque échantillons à l'onde courante. La commande Play saisira alors un instrument directement de la banque échantillons.

```
Load "SAMPLES/SAMPLES.Abk"
Sample 1 To 15
For I=20 To 50
      Play I,50
Next I
```

Comme de coutume, *voix* vous permet de sélectionner une gamme de voix définies par cette instruction. C'est un mode point standard dont la syntaxe est la suivante:

Bit 0-> Voix 0
Bit 1-> Voix 1
Bit 2-> Voix 2
Bit 3-> Voix 3

Chaque voix peut être sélectionnée en forçant le bit correspondant à 1.

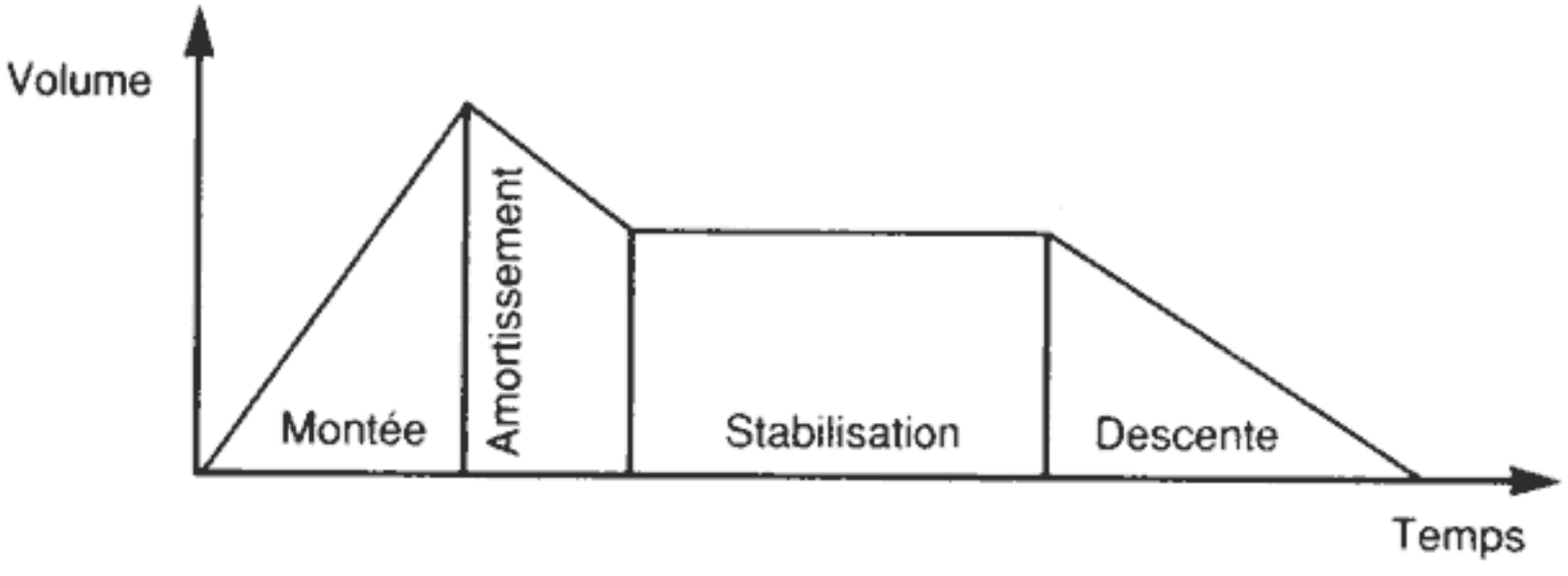
Remarque: La gamme de notes qu'un échantillon peut jouer dépend de sa vitesse d'enregistrement initiale. Si une note est trop élevée, il se peut qu'AMOS ne puisse pas du tout la jouer. La gamme acceptable varie selon les échantillons mais elle est d'habitude comprise entre 10 et 50.

SET ENVEL *(Créer une enveloppe)*

SET ENVEL onde,phase TO durée,volume

La commande SET ENVEL fait varier graduellement le volume de la note lorsque celle-ci est jouée. Dans le monde réel, les sons ne sont pas entièrement formés à leur émission. Leur volume change pendant un espace de temps en formant une courbe appelée enveloppe; cette enveloppe varie en fonction du type d'instrument que vous jouez. Voici un exemple typique de l'une de ces enveloppes.

Une enveloppe typique



L'émission de ce son est divisé en quatre phases: montée, amortissement, stabilisation et descente.

AMOS Basic vous permet de définir vos enveloppes en sept étapes distinctes. Chaque étape représente un changement régulier du volume de la note courante.

onde représente le numéro de l'onde qui sera affecté par cette instruction. Vous pouvez choisir toutes les formes d'ondes que vous souhaitez à cette fin, y compris les générateurs intégrés NOISE et SINE.

phase représente la durée de l'étape qui devra être définie entre 0 et 6.

durée spécifie la durée de l'étape courante en 50èmes de seconde. Elle determine la vitesse apparente du changement de volume lors de cette phase.

volume spécifie la portée du volume à la fin de cette phase. Les niveaux de volume autorisés sont compris entre 0 et 63.

Il est important de bien comprendre que ce volume se rapporte à l'intensité précédemment définie à l'aide de la commande VOLUME. Même si la note est basse, la forme de l'enveloppe est parfaitement conservée. Voici quelques exemples:

```
Set Envel 1,0 To 200,63 : Rem Fixe la durée de la première étape
Play 40,0 : Rem Cette note sera jouée pendant quatre secondes
```

Vous pouvez entendre que le volume de votre son commence de zéro et s'accroît pour atteindre son intensité maximale pendant la durée de la note. Essayons de définir quelque chose d'un peu plus compliqué:

```
Set Envel 1,0 To 15,60 : Rem Montée lente
Play 40,0 : Wait Key
Set Envel 1,1 To 1,50 : Rem Montée rapide
Play 40,0 : Wait Key
Set Envel 1,2 To 10,50 : Rem Courte stabilisation
Play 40,0 : Wait Key
Set Envel 1,3 To 50,0 : Rem Longue descente
Play 40,0
```

Finalement, voici un exemple d'une enveloppe NOISE:

```
Noise To 15
Set Envel 0,0 To 1000,30
Play 40,0
Wait key
Music Off : Rem Désactive le son
```

Ne confondez pas les ondes et les enveloppes. Une onde définit les éléments de fréquence de vos notes, alors qu'une enveloppe change leur volume selon une forme prédéfinie.

Synthétiseur de parole

Vous trouverez le programme d'un synthétiseur de parole puissant sur la disquette Workbench normale. A l'aide de cette routine, vous pouvez faire parler vos programmes

AMOS. Un synthétiseur de parole est particulièrement utile dans l'éducation puisque la plupart des jeunes répondront bien mieux à l'écoute de paroles qu'à la lecture d'un texte écrit ennuyeux.

SAY *(Prononcer des paroles)*

SAY t$[,mode]

La commande SAY est incroyablement facile à utiliser. Tapez votre phrase en la terminant par un signe de ponctuation comme un point. SAY traduira votre phrase en une forme interne et la restituera directement par vos enceintes. Exemple:

```
Say "AMOS AMOS AMOS"
```

Lorsque vous utilisez cette instruction pour la première fois, le dispositif NARRATOR se chargera automatiquement du disque. Il est donc important de s'assurer qu'une disquette est placée dans le lecteur courant avant de vous servir de ce système, sinon vous obtiendrez une boîte de dialogue de style Intuition.

mode échange les deux modes de parole. Par défaut, votre programme attendra que les paroles soient prononcées et les effets musicaux ou sonores seront temporairement interrompus. Si vous forcez *mode* sur un, vous activez un système multitâches vous permettant d'émettre vos paroles pendant qu'AMOS déroule votre programme. Bien évidemment, cela ralentira considérablement l'exécution de vos routines Basic. Pour réinitialiser le système, forcez *mode* à zéro. Exemple:

```
Do
   Input "Taper une phrase?";T$
   T$=T$+"."
   Say T$,1
Loop
```

Si le synthétiseur de paroles ne peut pas comprendre ce que vous essayez de dire, vous n'obtiendrez pas un message d'erreur mais l'exécution de la commande sera automatiquement suspendue.

Notez également que le le synthétiseur peut parfois être désorienté par les phrases très courtes. Le reste de la phrase précédente est quelques fois collée à celle qui est en train d'être prononcée. Vous pouvez résoudre le problème en ajoutant simplement quelques espaces à la fin de votre phrase. Ceux-ci effaceront les données vocales indésirées.

SET TALK *(Définir des effets vocaux)*

SET TALK sexe,mode,hauteur,vitesse

Cette commande vous permet de modifier le type de voix qui sera utilisée par la commande SAY.

sexe choisit entre une voix masculine (0) ou féminine (1). En toute honnêté, la restitution n'est pas particulièrement réaliste. Vous pouvez obtenir de meilleurs effets en augmentant simplement la fréquence de la voix au moyen du paramètre hauteur.

mode donne un mouvement rythmique étrange à la voix. Vous pouvez activer mode en le forçant à une valeur de 1. L'effet rendu est bien agréable mais pas très humain. Vous pouvez rétablir la voix sur normal avec un paramètre de zéro.

hauteur modifie la fréquence de la voix, de 65 (basse) à 320 (très haute).

vitesse spécifie la vitesse, mesurée en mots par minute. Les vitesses de débit de paroles permises sont comprise entre 40 et 400.

N'importe quel des paramètres ci-dessus peuvent être omis si besoin. A condition que vous placiez les virgules à leurs positions normales, vous pouvez changer les jeux d'options de façon indépendante. Voici quelques exemples:

```
Set Talk 1,,, : Say "C'est une voix féminine."
Set Talk,1,,, : Say "C'est une voix féminine avec intonation."
Set Talk 0,0,320, : Say "Une voix soprano."
Set Talk,,65 : Say "Une voix grave."
Set Talk 0,0,100,400 : Say " Dites ceci rapidement."
Set Talk ,,,40 : Say "C'est lent."
```

Les effets filtre

LED *(Activer un filtre passe-haut/LED d'allumage)*

LED ON/OFF

La commande LED a deux actions bien distinctes. Elle permet non seulement d'activer ou de désactiver la led d'allumage de la console de l'Amiga, mais elle contrôle également un filtre *passe-bas* spécial.

Le filtre change la façon dont les sons de haute fréquence sont traités par le système. Normalement, ces sons sont filtrés de manière à éviter des effets de distortion indésirés. Malheureusement, ceci ôte le timbre à de nombreux instruments à percussion. En désactivant le filtre, vous pouvez retrouver la qualité essentielle de ces instruments. Exemple:

```
Load "AMOS_DATA:MUSIC/MUSIC.ABK"
Music 1
Do
   If mouse key=1 Then Led on
   If mouse key=2 Then Led off
Loop
```

En faisant des essais avec cette fonction, vous vous apercevrez que certains sons musicaux sont de meilleure qualité sans le filtre, alors que d'autres sont déformés d'une façon horrible si le filtre est désactivé. La led d'allumage vous permet de voir rapidement la position du filtre courant. En modulant la led à temps dans la musique, vous pouvez considérablement améliorer l'effet global.

18 Le clavier

Dans un avenir lointain, nous taperons sans doute nos programmes directement par une certaine forme de pensée. Avant ce temps là, la façon la plus rapide d'entrer des informations dans un ordinateur est de les taper au clavier.

AMOS Basic met à votre disposition des douzaines de commandes utiles du clavier. Vous pouvez les utiliser à n'importe quelle fin, du jeu d'arcade au jeu d'aventures. Il est même possible d'écrire un traitement de texte complet uniquement en AMOS Basic!

=INKEY$ *(Obtenir l'appui d'une touche)*

k$=INKEY$

La fonction INKEY$ vérifie si l'utilisateur a appuyé sur une touche et renvoie sa valeur dans la chaîne *k$*.

Notez que la commande INKEY$ n'attend pas que vous entriez des données. Si vous n'avez pas tapé de caractères, INKEY$ renverra simplement une chaîne vide "". Exemple:

```
Do
   Rem Essayez de frapper quelques-unes des touches fléchées
   X$=Inkey$: If X$<>"" Then Print X$;
Loop
```

La commande INKEY$ est seulement capable de lire les touches renvoyant un caractère spécifique ASCII du clavier. Ascii est un code standard utilisé pour représenter tous les caractères pouvant être imprimés à l'écran.

Il est important de bien comprendre que certains touches, comme le bouton HELP ou les touches de fonction, utilisent un format plutôt différent. Si INKEY$ détecte une telle touche, elle renverra un caractère de valeur zéro (CHR0)). Vous pouvez alors trouver le code de position de la touche en utilisant une fonction indépendante SCANCODE.

SCANCODE *(Entrer le code de position de la dernière touche elle-même entrée à l'aide de INKEY$)*

s=SCANCODE

La commande SCANCODE renvoie le code de position interne d'une touche qui a été précédemment entrée à l'aide de la fonction INKEY$. Ceci vous permet de rechercher les touches du clavier ne produisant pas de caractères, comme les touches HELP ou TAB. Tapez le petit exemple suivant:

```
Do
   While K$=""
        K$=Inkey$
   Wend
   If Asc(K$)=0 Then Print "Vous avez appuyé sur Une Touche Sans Code ASCII."
   Print "Le Code de position est";Scancode
   K$=""
Loop
```

=KEY STATE *(Vérifier si une touche a été enfoncée)*

t=KEY STATE(s)

Cette commande vérifie si une touche spécifique du clavier de l'Amiga a été enfoncée. s est le code de position interne de la touche que vous voulez vérifier. Si vous êtes en train d'appuyer sur cette touche, alors KEY STATE renverra la valeur -1 (vrai), sinon le résultat sera FAUX (0). Exemple:

```
Do
   If Key State(69)=True Then Print "Echappé!" : Rem Esc Touche enfoncée
   If Key State(95)=True Then Print "HELP!" : Rem Touche HELP enfoncée
Loop
```

=KEY SHIFT *(Renvoyer l'état des touches majuscules)*

keys=KEY SHIFT

Cette commande renvoie l'état courant des diverses touches de commande. Ces touches telles que MAJUSCULES ou Alt ne peuvent pas être détectées en utilisant les commandes INKEY$ ou SCANCODE. Mais vous pouvez facilement tester les combinaisons des touches de commandes par un simple appel à la fonction KEY SHIFT. *keys* est un mode point dont la syntaxe est la suivante:

Bit	Touches testées	Notes
0	Touche MAJUSCULE de gauche	
1	Touche MAJUSCULE de droite	
2	Touche de verrouillage majuscule	Soit ACTIVEE ou DESACTIVEE
3	Control (Cntrl)	
4	Alt de gauche	
5	Alt de droite	
6	Amiga de gauche	C'est la touche Commodore sur certains claviers
7	Amgia de droite	

Si un bit est forcé à un, cela signifie que la touche correspondante a été enfoncée.

Exemple:

```
Centre "Appuyez sur quelques touches de commande"
Curs Off
Do
   Locate 14,4 : Print Bin$(Key Shift,8)
Loop
```

=INPUT$(n) *(Entrer n caractères dans une chaîne)*

x$=INPUT$(n)

Cette commande permet d'entrer n caractères, l'un après l'autre, directement du clavier. Comme avec INKEY$, ces caractères ne sont pas renvoyés à l'écran.

x$ est une variable chaîne qui recevra vos nouveaux caractères. *n* représente le nombre de caractères à entrer. Exemple:

```
Clear Key : Print "Tapez Dix Caractères"
C$=Input$(10) : Print "Vous avez entré ";C$
```

Cette instruction est **différente** de la commande classique INPUT. Les deux instructions sont complètement différentes. Notez également qu'il existe une version spéciale de la commande INPUT$ pouvant être utilisée pour lire vos caractères depuis le disque.

WAIT KEY (Attendre l'appui d'une touche)

WAIT KEY

Cette instruction attend un seul appui d'une touche. Exemple:

```
Print "Appuyez Sur Une Touche" : Wait Key : Print "Touche Enfoncée"
```

KEY SPEED *(Changer la vitesse de frappe)*

KEY SPEED attente,vitesse

Cette commande vous permet d'adapter la vitesse de frappe des touches du clavier à votre convenance. La nouvelle vitesse s'applique à toutes les parties du système AMOS, y compris l'éditeur.

attente est la durée en 50èmes de seconde entre l'appui d'une touche et le début de la séquence de répétition.

vitesse est la durée en 50èmes de seconde entre chaque caractère successif. Exemple:

```
key$(1)="Key Speed 10,1"+Chr$(13) : Rem Sauvegarder une vitesse raisonnable
Key Speed 10,10 : Rem Abaissez une touche pour une répétition LENTE
```

```
Key Speed 1,1 : Rem Répétition RAPiDE
Rem Appuyez sur la touche Amiga de gauche F1 pour réinitialiser votre clavier
```

CLEAR KEY *(Initialiser le tampon clavier)*

CLEAR KEY

Lorsque vous entrez un caractère du clavier, son code Ascii est placé dans une zone de mémoire connue sous le nom de tampon clavier. C'est ce tampon qui est examiné par la fonction INKEY$ pour obtenir les touches pressées.

CLEAR KEY efface complètement le tampon et réinitialise votre clavier. Cette commande est particulièrement utile au commencement d'un programme, puisqu'il se peut que le tampon soit rempli d'informations indésirées. Vous pouvez également l'appeler immédiatement avant une commande WAIT KEY pour vous assurer que le programme attende le nouvel appui d'une touche avant de se dérouler. Exemple:

```
Clear key : Rem supprimer les abaissements courants de touche
Wait key : Rem Attendre le nouvel appui d'une touche
```

PUT KEY *(Introduire une chaîne dans le tampon clavier)*

PUT KEY a$

Cette commande charge une chaîne de caractères directement dans le tampon clavier. Vous pouvez y intégrer un retour du curseur au moyen du caractère CHR$(13) (RETOUR).

PUT KEY est la plus couramment utilisée pour définir la configuration des routines INPUT. Voici une démonstration:

```
Do
   Put Key "Non"
   Input "Un autre jeu ";A$
   If A$="Non" Then Exit
Loop
```

Le programme ci-dessus affecte un "Non" à la chaîne implicite INPUT. Cette valeur sera entrée directement dans la variable A$ au moyen de la touche RETOUR. Vous pouvez également éditer la ligne en utilisant les touches fléchées et entrer votre propre valeur dans A$ si besoin.

Entrée/Sortie

INPUT *(Charger votre propre valeur et l'entrer dans la variable)*

Cette commande vous permet d'entrer des informations dans une ou plusieurs

variables. Il existe deux syntaxes possibles de cette instruction:

INPUT vars[;]

Cette commande saisit une liste de variables du clavier. *var* peut comprendre n'importe quel jeu de variables que vous souhaitez, séparé par des virgules. Un point d'interrogation s'affichera automatiquement à la position courante du curseur.

INPUT "Prompt";variable list[;]

Cete commande affiche à l'écran la chaîne *prompt* avant d'entrer vos informations. Notez que vous devez toujours placer un point-virgule entre votre texte et la liste de variables. Vous **n'êtes pas** autorisé à utiliser une virgule à cette fin.

Le point virgule en option ";" à la fin de votre liste de variables spécifie que le curseur du texte ne sera pas affecté par l'instruction INPUT et gardera sa position d'origine après que les données aient été entrées.

Lorsque vous exécutez une de ces commandes, le Basic attend que vous entriez les informations nécessaires à l'aide du clavier. Chaque variable de votre liste doit correspondre à une seule de vos valeurs. Celles-ci doivent être du même type que les variables d'origine et être séparées par des virgules. Voici quelques exemples simples:

```
Input A
Print "Votre numéro était";A
Input "Entrez un nombre à virgule flottante";N#
Print "Vous avez entré";N#
Input "Quel est votre nom, Humain?";name$
Locate 23, : Print "Bonjour ";Name$
```

Voir INPUT# et LINE INPUT

LINE INPUT *(Entrer une liste de variables séparées par un Retour)*

INPUT "Prompt";variable list[;]
INPUT var[;]

Cette commande est tout à fait identique à INPUT, sauf qu'elle utilise un Retour au lieu d'une virgule pour séparer chaque valeur que vous tapez au clavier. Exemple:

```
Line Input "Entrez trois nombres";A,B,C
Print A,B,C
```

Voir INPUT, LINE INPUT#

19 Autres commandes

Comme toutes les versions du langage Basic, AMOS comprend également une panoplie de commandes banales telles que PRINT et DATA. Comme vous allez le découvrir dans ce chapitre, AMOS a ajouté de l'inédit à ces instructions. Donc même les parties les plus ennuyeuses de l'AMOS Basic sont tout à fait intéressantes!

PRINT ou ? *(Imprimer une liste de variables à l'écran)*

PRINT items

L'instruction PRINT affiche des informations à l'écran, à partir de la position courante du curseur.

La liste d'articles peut comprendre n'importe quel groupe de variables ou de constantes que vous souhaitez, à condition que vous ne dépassiez pas la longueur maximale de ligne permise (255).

Chaque élément de votre liste doit être séparé par un point virgule ";" ou une virgule ",". Un point-virgule permet d'imprimer les données immédiatement après la valeur précédente, alors qu'une virgule déplace d'abord le curseur sur la position TAB suivante à l'écran.

Normalement, le curseur sera déplacé d'une ligne vers le bas après chaque instruction PRINT. Vous pouvez supprimer cet effet en ajoutant un caractère de séparation après l'impression. Comme auparavant, un point-virgule marque la position du curseur après l'opération et une virgule place le curseur sur l'arrêt du TAB suivant avant d'exécuter la prochaine opération.

```
Print "C'est L'Histoire Du Guide Des Auto-Stoppeurs Vers La Galaxie"
A=10 : B=20 : C$="Trente": Print A,B;C$
Print 10,20*10,"Enfer";
Print "vers"
```

Voir également USING, LPRINT and PRINT#

USING *(Sortie formatée)*

PRINT USING format$;variable list

L'instruction USING est utilisée avec PRINT pour permettre un contrôle plus précis du format de votre listage.
format$ spécifie une liste de caractères définissant la façon dont les variables seront affichées à l'écran. Tous les caractères normaux de cette chaîne seront directement affichés mais si vous y intégrez un des caractères ~ # + - . ; ^ alors une parmi les nombreuses et utiles opérations de formattage sera exécutée.

~ Formatte une variable en chaîne. Chaque ~ est remplacé par un seul caractère de la chaîne sortie, prise de gauche à droite.

Print Using "C'est une ~~~~~~ démonstration de USING";"Petite"
C'est une petite démonstration de USING.

Chacun des dièzes spécifie un seul chiffre de votre variable. N'importe quels des chiffres inutilisés de cette liste sera automatiquement remplacé par des espaces.

Print Using "####";314211
4211

\+ Ajoute un signe positif à un nombre s'il est positif et un signe minuscule s'il est négatif.

Print Using "+##";10: Print Using " +##";-10
+10
-10

\- Intègre seulement ce signe si le nombre est négatif. Les nombres positifs sont précédés d'un espace.

Print Using "-##";10 : Print Using "-##";-10
10
-10

. Le point place un point décimale dans un nombre et le centre parfaitement à l'écran.

Print Using "PI est #.###";3.1415926
PI est 3.141

; Centre un nombre sans afficher le point décimale

Print Using "PI est #;###";PI#
PI est 3 141

REM ou ‘ *(Remarque)*

REM comment

L'instruction REM est utilisé pour ajouter des commentaires à votre programme Basic. AMOS Basic ne tiendra pas du tout compte des libellés que vous tapez après une instruction REM. Exemple:

Rem Ce programme ne fait absolument rien
‘C'est un commentaire

L'instruction normale REM peut être utilisée pratiquement à tout point de votre programme Basic. Mais vous pouvez seulement placer une apostrophe ' au tout début de l'une de vos lignes. Exemples:

```
' Un simple commentaire
Print "Enfer" : Rem C'est ok
Print "Au revoir" : 'Ceci produira une erreur
```

DATA *(Placer une liste de données élémentaires dans un programme AMOS Basic)*

DATA liste d'articles

L'instruction DATA vous permet d'intégrer des listes entières d'informations utiles directement dans un programme Basic. Vous pouvez ultérieurement charger les données dans une ou plusieurs variables en utilisant l'instruction READ. Chaque variable de votre liste est séparée par une seule virgule. Exemple:

```
Data 1,2,3,"Bonjour"
```

A la différence des autres Basics, la version AMOS de cette instruction vous permet également d'intégrer des expressions formant une partie de vos données. Ainsi les lignes suivantes de programmation sont aussi bien acceptables:

```
Data $FF50,$890
Data %1111111111111,%1101010101
Data A
Label: Data A+3/2.0-Sin(B)
Data "Salut" + "Les gars"
```

Il est important de bien comprendre que "A" à LABEL est entré en tant que contenu de la variable A et non comme le caractère A. L'expression sera calculée automatiquement pendant l'opération READ en utilisant les toutes dernières valeurs de A et B.

Notez également qu'il ne doit y avoir qu'une instruction DATA dans la ligne courante. Tout ce qui se trouve après cette commande sera complètement rejeté! Vous pouvez placer les instructions de données où vous voulez dans votre programme Basic. Toutefois, les données que vous enregistrez et AMOS sont seulement accessibles depuis le programme principal. Chaque procédure peut avoir son propre jeu d'instructions DATA, ces dernières étant complètement indépendantes du reste de votre programme. Voici une démonstration:

```
TEST : Read A$ : Print A$
Data "Données du programme"
Procedure Test
   Read B$ : Print B$
   Data "Données de la procédure"
```

```
End proc
```

Voir READ, RESTORE.

READ *(Lire les données d'une instruction DATA dans une variable)*

READ list of variables

L'instruction READ charge des informations mémorisées dans une instruction DATA dans une liste de variables. READ utilise un repère spécial pour déterminer l'emplacement de la prochaine donnée qui doit être lue. Au lancement de votre programme, le repère est déplacé sur la donnée élémentaire suivante de la première instruction DATA. Une fois que cette donnée a été lue, le repère est avancé et indique la donnée suivante de votre liste. Comme vous pouvez le supposer, les variables que vous lisez doivent être exactement du même type que les données se trouvant à la position courante. Exemple:

```
T=10
Read A$,B,C,D$
Print A$,B,C,D$
Data "Chaîne",2,T*20+rnd(100),"AMOS"+"Basic"
```

Voir RESTORE,DATA.

RESTORE *(Modifier l'indicateur courant de READ)*

RESTORE Label
RESTORE LABEL$
RESTORE ligne
RESTORE nombre

L'instruction RESTORE change le point auquel une opération READ ultérieure suppose trouver l'instruction DATA suivante. Chaque procédure AMOS a son propre indicateur de données. Ainsi, les appels à cette commande s'appliqueront seulement à la procédure **courante!**

label et une étiquette spécifiant la position de la première instruction DATA à lire. Le nom de cette étiquette peut être calculé comme une partie d'une expression. Ainsi les commandes Basic suivantes sont toutes parfaitement autorisées:

```
Restore L
Restore "L"+"A"+"B"+"E"+"L"
```

Similairement, ligne sélectionne le numéro de ligne de l'instruction suivante DATA. Comme label, elle peut être entrée comme une expression:

```
Restore 10
```

```
Restore TEST+2
```

Tout en vous permettant de faire des branchements à volonté dans les instructions DATA de votre programme, RESTORE vous laisse choisir vos informations en fonction de vos actions. La description de chaque lieu dans un jeu d'aventure pourrait être par exemple rangée dans un bloc de simples instructions DATA. Pour lire cette description, vous pourriez utiliser le programme suivant:

```
Restore ROOM*5+1000 : Rem Chaque LIEU a 5 ordres de données
Read DESC$ : Print DESC$
1000 Data "Description de Lieu 1"
1005 Data "Texte de Lieu 2"
1010 Data "Lieu 3"
: :   :
```

Bien évidemment, si un ordre de données n'existe pas à la ligne spécifiée par RESTORE, un message d'erreur sera produit. Faites attention en essayant d'utiliser cette commande à l'intérieur d'une procédure. Afin de pouvoir s'exécuter, vos instructions DATA DOIVENT se trouver à l'intérieur de la procédure courante.

Voir également READ, DATA

WAIT *(Attente en 50èmes de seconde)*

WAIT n

Cette instruction suspend un programme AMOS Basic pendant n 50ème de seconde. Toutes les fonctions utilisant des interruptions, telles que MOVE et MUSIC continueront à fonctionner normalement pendant cette attente. Exemple:

```
Wait 50
```

Cette instruction indique une attente d'une seconde.

=TIMER= *(Comptage en 50èmes de seconde)*

v=TIMER
TIMER=v

La fonction TIMER est une variable réservée qui est augmentée d'une unité chaque 50ème de seconde. Elle est couramment utilisée pour définir la valeur de départ d'un générateur de nombres aléatoires comme suit:
Randomize Timer

MULTI WAIT *(Forcer une attente multi-tâche vbl)*

MULTI WAIT

Pour réaliser de vrais programmes multi-tâches, vous ne devez pas saisir tout le temps du processeur et garder de la puissance pour les autres tâches. MULTI WAIT produit une attente MULTI-TASK vbl. Vous devez l'utiliser dans la boucle principale de votre programme, lorsque vous attendez que l'article d'un menu soit sélectionné par exemple. Notez que vous ne devez pas utiliser cette instruction pour réaliser une synchronisation précise d'écran. Puisqu'elle est conçue pour la multiprogramation, cette instruction n'est pas du tout précise! Elle peut sauter de nombreuses VBLs, en fonction du nombre de tâches en cours d'exécution.

Si elle vous a échappé au cours de la lecture du manuel, la multiprogrammation peut être activée en appuyant sur Amiga+A pour passer d'AMOS aux environnements du CLI ou du WorkBench. Ceci permet aux systèmes d'une puissance d'au moins 1 mega de mémoire de faire fonctionner à la fois AMOS et les programmes DPAINT III!

NOT *(Opération logique NON)*

v=NOT(d)

Cette fonction change chaque chiffre binaire en un 1 et un 0 et vice versa. Puisque True=-1 (%1111111111111) en binaire et False=0, NOT(True)=False. Exemple:

```
Print Bin$(Not(%1010),4)
```

%0101

```
If Not(True)=False Then Print "Faux"
```

TRUE *(VRAI logique)*

v=TRUE

Si vous effectuez un test tel que X>10, une valeur est produite. Si l'état est vrai, alors ce nombre est forcé à -1, sinon il sera de zéro.

```
If -1 Then Print "Moins 1 est VRAI"
If TRUE Then Print "et VRAI est";True
```

Voir FALSE,NOT

FALSE *(FAUX logique)*

v=FALSE
Cette fonction renvoie une valuer de zéro. Elle est utilisée par toutes les opérations conditionnelles telles que IF...THEN et REPEAT...UNTIL pour représenter FALSE.

```
Print False
0
```

Voir TRUE.

AMOS TO BACK *(Cacher AMOS et afficher le Workbench)*

AMOS TO BACK

Ceci amène le Workbench en premier plan à l'écran et vous permet d'accéder à d'autres programmes.

AMOS TO FRONT *(Ramener AMOS à l'écran)*

AMOS TO FRONT

AMOS est ramené à l'écran grâce à cette commande et le Workbench est caché derrière.

AMOS HERE *(Editer le type de tâche à l'écran)*

AMOS HERE

Cette commande renvoie VRAI si AMOS est affiché et FAUX si le Workbench est visualisé.

20 Accès au disque

Les commandes du disque AMOS vous donnent un accès à l'intégralité du système de fichier de l'Amiga. Elles peuvent être exploitées pour créer tout ce que vous voulez, du lecteur simple de fichier à la base de données complète.

Pour ce qui est de sélectionner vos fichiers, AMOS est particulièrement impressionnant. Il existe une routine pré-définie de sélecteur de fichier vous permettant de choisir vos noms de fichier depuis une boîte de dialogue. Elle est très facile à utiliser et ajoute une note vraiment professionnelle à vos programmes Basic.

AMOS peut également lire les répertoires, effacer des fichiers, créer des dossiers directement depuis un de vos programmes. Il y a même une commande qui vous permet de rechercher l'existence d'un fichier précis sur la disquette!

Enfin, nous vous prêtons un support spécial si vous possédez déjà le progiciel STOS Basic initial. AMOS est tout à fait compatible avec le puissant système CROSS DOS de CONSULTRON. Donc, si vous avez acheté ce produit, il vous sera facile d'importer vos programmes STOS Basic directement dan l'AMOS Basic.

Lecteurs et volumes

Comme vous le savez, l'Amiga vous permet d'étiquetter vos disquettes de bien différentes manières. Si vous êtes peu familier avec le CLI, vous allez peut-être trouver certains termes un peu déroutants. Nous allons donc vous donner une brève explication des diverses termes conventionnels.

Lecteurs

Chaque lecteur branché sur l'Amiga se rapporte à un code d'identification standard comportant trois lettres. Afin de distinguer ce code d'un nom de fichier normal, on lui ajoute le signe deux points ":" lorsqu'il est intégré aux instructions Basic.

Lecteurs de disquette: Sont affectés les noms dans la syntaxe suivante:

Dfn:

n représente un seul chiffre représentant le numéro de votre lecteur. Le premier lecteur de disquette de votre système (habituellement le lecteur interne) est connu sous le nom de Df0: puis viennent les lecteurs Df1:,Df2: et Df3: s'ils sont installés.
Lecteurs de disque dur: Ceux-ci sont spécifiés en utilisant:

Dhn:

où n est le numéro du lecteur de votre disque dur.

Volumes

L'Amiga donne également un nom VOLUME particulier à chaque disquette. Cette étiquette peut être remplacée par le nom du lecteur dans n'importe quelle de vos commandes AMOS Basic. AMOS recherchera automatiquement chaque lecteur disponible. S'il ne le trouve pas, vous obtiendrez un message *d'erreur Lecteur non installé.*

Si vous préparez une nouvelle disquette du Workbench, la disquette sera affectée du nom "Empty". Pour changer cette étiquette du Workbench, il vous suffit de cliquer sur l'option RENOMMER et de taper votre nouveau nom dans la boîte de dialogue prévue. Ce nom peut représenter toute chaîne de caractères que vous voulez mais il doit être suivi d'un signe deux points si vous souhaitez l'utiliser dans vos programmes, un peu comme le nom du lecteur. Voici quelques noms typiques de volume:

AMOS:
AMOS_DATA:

Ceux-ci pourraient être utilisés de la façon suivante:

```
Load "AMOS: Sprites.Acc" Rem Charger l'éditeur de sprite depuis une disquette appelée AMOS

Dir "AMOS_DATA:" : Rem Saisir le répertoire de AMOS_DATA:
```

Attention! Si vous donnez le même nom à plusieurs disquettes ou si vous les échangez au hasard, l'Amiga peut facilement ne plus savoir à quelle disquette vous vous rapportez. Dans ce cas, il vous faudra taper le nom du lecteur. AMOS sera ainsi précisément informé de la location du disque recherché dans votre système. Exemple:

```
Dir "Df0:"
```

Nous vous conseillons fortement d'affecter un nom différent à chaque disquette que vous utilisez. Cela prend seulement quelques secondes depuis le Workbench et simplifie beaucoup les choses.

Unités logiques

Enfin, il existe également un ensemble d'objets connus sous le nom d'unités logiques. Elles sont utilisées par les routines d'exploitation de l'Amiga pour déterminer la position précise des fichiers système importants tels que les programmes de gestion des unités ou les polices. Chaque unité est normalement affectée à un répertoire spécifique sur la disquette courante de mise en route. Voici quelques exemples utilisés par AMOS.

POLICES:	Un répertoire contenant les polices courantes.
LIBS:	Contient un fichier bibliothèque demandé par la commande AMOS Say

Tapez la ligne suivante depuis le mode direct:

```
Dir "LIBS:"
```

Cross Dos

Si vous avez acheté le progiciel CROSS DOS et l'avez installé en mémoire, il vous sera possible d'accéder à des disquettes de format IBM ou ST dans l'AMOS Basic. Ces disquettes sont affectées d'un nom commençant par les lettres DI:

DIn: (où n est le numéro de votre lecteur)

Afin de convertir vos programmes STOS en AMOS Basic, il vous faudra les sauvegarder en format ASCII au moyen de l'option FSAVE "*.ASC" de STOS. Insérez ensuite la disquette dans un lecteur de disquette de l'Amiga qui a été installé par Cross-Dos en tant que lecteur IBM.

Remarque: En raison des différences existant entre AMOS et STOS, il vous faudra modifier légèrement la plupart des programmes STOS avant de les faire fonctionner sous AMOS. Si vous voulez utiliser les fonctions spéciales de l'AMOS Basic, vous serez peut être obligé de modifier considérablement certains programmes. Toutefois, cela vaut toujours la peine de convertir vos programmes STOS en AMOS. La puissance supplémentaire du hardware de l'Amiga peut transformer vos jeux STOS au point de les rendre méconnaissables!

Changement de répertoires

DIR *(Afficher le répertoire du disque courant)*

DIR [CHEMIN$][/W]

Le répertoire donne une liste de tous les fichiers existant sur le disque courant. Si le chemin$ en option est spécifié, seuls les fichiers satisfaisant un certain nombre de conditions seront affichés. Tous les dossiers de la liste seront différenciés par le signe "*" placés devant leur nom.

La liste peut être interrompue à tout moment en appuyant sur la barre d'espacement. Pour le reprendre, il vous suffit d'appuyer à nouveau sur cette barre.

Notez que si vous changez de disquette et essayez de rechercher le directory, vous obtiendrez un *message d'erreur Unité non installée*. Ceci s'explique par le fait que vous avez retiré le disque courant sans informer l'AMOS Basic. Il vous suffit alors d'actualiser le nom du directoire courant du nouveau disque au moyen d'une ligne comme par exemple: DIR$=Df0:" sans appeler DIR.

/W liste les fichiers sur deux colonnes à l'écran. Cela permet de doubler le nombre de fichiers à l'affichage.

La chaîne du chemin comprend trois éléments principaux:

[Disque:][Répertoire/] Filtre

Disque C'est le nom du disque à examiner. Afin d'éviter que le nom du fichier soit interprété comme un fichier normal, il vous faudra le faire suivre de deux points ":". Exemple:

```
Dir "AMOS:"
Dir "Df0:"
Dir "FONTS:"
```

Répertoire/ Représente le nom d'un simple dossier que vous souhaitez afficher. Exemples:

```
Dir"AMOS_DATA:IFF/"
Dir "IFF/"
```

Filtre Définit un ensemble de conditions devant être satisfaites pour chaque fichier de votre listage
Texte Normal: Chaque caractère de votre texte doit correspondre exactement à un seul caractère du nom du fichier à afficher. Exemple:

```
Dir "MUSIC"
Music
```

**(astérisque):* Permet de faire une recherche de concordance de n'importe quel groupe de lettres de vos noms de fichiers jusqu'à l'exécution du caractère de commande suivant. Exemple:

```
Dir "M*" : Rem Répertorier tous les fichiers dont les noms commencent par la lettre M
Musique
Mercure
Malt
```

Par défaut, cette option ne tiendra pas compte des fichiers comprenant un suffixe de type MS-DOS. Ainsi, un fichier du disque comme *Mad.Asc* ne sera pas répertorié.

.(point): Permet de faire une recherche de concordance du suffixe du nom d'un fichier. Il est couramment utilisé avec la commande "*" pour afficher tous les fichiers comportant un suffixe particulier. Voici quelques exemples:

```
Dir ".*" : Rem Répertorier tous les fichiers de la disquette comportant un suffixe
Dir "*.Amos" : Rem Répertorier tous les fichiers AMOS
Dir "Programme.*" : Rem Affiche tous les noms commençant par Programme
```

?: Permet de faire une recherche de concordance d'un seul caractère à la position courante. Ainsi **Dir "AM??"** répertorie les fichiers comme:

```
AMOS
AMAL
```

Toutefois, il ne tiendra pas compte d'un nom comme "AMOS-BASIC" parce qu'il contient plus de quatre caractères. Chaque lettre du nom du fichier doit correspondre à un seul caractère de la chaîne de recherche afin que le fichier puisse s'afficher.

Voir LDIR

=DIR$ *(Changer le répertoire courant)*

s$=DIR$
DIR$=s$

DIR$ contient le répertoire qui sera utilisé comme point de départ de toutes les actions ultérieures, telles que le chargement et la sauvegarde. Elle est très similaire à la commande CD du CLI, avec l'avantage supplémentaire qu'elle vous permet aussi bien de lire le répertoire que de le changer. Exemple:

```
Print Dir$ : Rem Afficher le répertoire courant
Dir$="AMOS_DATA:IFF/" : Rem Affecter le répertoire au dossier IFF de la disquette AMOS
```

Comme la fonction CD qu'elle remplace, tous les répertoires sont supposés se rapporter à celui que vous êtes en train d'utiliser. Tapez la ligne suivante par exemple:

```
Dir$="MANUAL/"
```

ou

```
Dir$="DF0:FONTS/"
```

PARENT *(Définit le chemin courant d'un seul répertoire)*

Le système de classement de l'Amiga vous permet d'emboîter les répertoires les uns dans les autres. Cela vous permet de classer facilement vos fichiers en plusieurs catégories. Prenons un petit exemple:

```
DOSSIERA
    DOSSIERB
        DOSSIERC/
    DOSSIERD/
```

Le schéma ci-dessus représente quatre dossiers distincts du disque. DOSSIERA est rangé dans le répertoire principal ou répertoire racine et comprend DOSSIERB et DOSSIERD. En jargon informatique, DOSSIERA est le père de DOSSIERC.

Comme vous pouvez l'imaginer, il est extrêmement facile de se perdre au cours de l'utilisation de ces dossiers. L'action de PARENT est de charger le répertoire courant avec le père du présent dossier. En utilisant cette commande à plusieurs reprises, vous pouvez rapidement retourner à votre répertoire racine d'origine. Exemple:

```
Dir$="\IFF"
Dir
Parent
Dir
```

SET DIR *(Définir le style utilisé par DIR)*

SET DIR n[,filtre$]

Cette instruction définit le style des listages de vos repertoires. *n* est le nombre de caractères compris entre 1 et 100 que contiendra chaque nom de fichier. Notez que cette définition **n'a pas** d'effet sur la longueur réelle de vos noms. Elle change seulement la façon dont ils seront affichés à l'écran.

filtre représente une liste des noms de chemin devant être **exclus** de vos recherches de répertoire. Tous les noms de fichier qui correspondent à ce filtre sont complètement rejetés et ne feront pas partie de votre répertoire à l'affichage. Cette instruction peut servir à supprimer les fichiers ennuyeux ".INFO" contenant les définitions d'icônes utilisées par le Workbench.

Notez qu'il est tout à fait possible d'ignorer une liste entière de chemins de fichier. Il vous suffit de faire suivre chaque nom d'une seule barre de fraction "/". Par défaut, le filtre est ainsi défini:

".INFO/*.INFO/*.*.INFO"

Exemple:

```
Set Dir 5 : Rem Afficher seulement les cinq premiers caractères de vos noms
Dir
Set Dir 30,"" : Rem Pas de filtre
Dir
```

Opérations courantes sur disque

=DFREE *(Espace disponible du disque)*

f=DREE

Cette commande renvoie l'espace disponible en octets, se trouvant sur le disque courant.

```
Print Dfree
```

MKDIR *(Créer un dossier)*

MKDIR f$

Cette commande crée un nouveau dossier appelé f$ sur le disque. Exemple:

```
Mkdir " Df0:TEST"
Dir
```

KILL *(Effacer un fichier du disque)*

KILL f$

Cette commande efface le fichier f$ du disque courant. **Attention!** Tout ce que vous effacez à l'aide de cette commande sera perdu!

RENAME *(Renommer un fichier)*

RENAME ancien$ TO nouv$

Cette commande change le nom d'un fichier. Si un fichier possède déjà le nom que vous venez de choisir, vous obtiendrez un message d'erreur.

Sélection d'un fichier

=FSEL$ *(Sélectionner un fichier)*

f$=FSEL$(chemin$[,défaut$][,titre1$, titre2$])

La fonction FSEL$ vous permet de choisir vos fichiers directement depuis le disque, en utilisant le sélecteur standard de fichier AMOS.

chemin$ définit le type de chemin de recherche déterminant les fichiers qui seront affichés dans votre listage.

Après avoir sélectionné un fichier, FSEL$ renverra un nom de fichier en entier ou une chaîne vide "" si vous avez sélectionné SORTIR.

défaut$ choisit un nom de fichier utilisé par défaut. Il sera automatiquement sélectionné en appuyant sur Retour.

titre1$* et *titre 2$ sont des chaînes de caractères optionnelles décrivant un titre à afficher en haut du sélecteur de votre fichier. Exemple:

```
f$=Fsel$("*.IFF","","Charger un fichier IFF")
If F$="" Then Edit : Rem Revenir sur l'éditeur si aucun fichier n'est choisi
Load IFF F$,0 : Rem Charger Fichier
```

Lancer un programme AMOS du disque

RUN *(Exécuter un programme AMOS Basic)*

RUN [fichier$]

Bien qu'il soit assez facile d'exécuter vos programmes AMOS directement depuis l'éditeur, nous avons également prévu une autre commande RUN. Cette version sans *fichier$* peut être seulement utilisée en mode direct.

Mais l'instruction RUN *fichier$* peut être également intégrée à un programme Basic.

Cela vous permet d'enchaîner une liste de programmes. Notez que si vous lancez un programme de cette façon, le programme existant sera supprimé de la mémoire et toutes les variables seront perdues. Les pages de données que vous avez créées resteront intactes permettant ainsi aux écrans de chargement intermédiaires de s'afficher. Exemple:

```
Print "Exécuter Le Test"
Run "Salut.AMOS " : Rem Sur La Disquette De Programme AMOS
Print "Cette Ligne N'est Jamais Exécutée"
```

Cette commande est extraordinairement utile, puisqu'elle vous permet de diviser un jeu AMOS en plusieurs niveaux pouvant être chargés séparément du disque. Chaque niveau peut être un programme complètement indépendant. Ainsi, la seule limite imposée à la taille de vos jeux est la capacité de rangement du disque! Vous pouvez donc produire des jeux géants grâce à ce système!

Voir également PRUN

Vérifier l'existence d'un fichier

=EXIST *(Vérifier si un fichier spécifique existe)*

flag=EXIST(F$)

Cette fonction permet de rechercher le fichier f$ dans le répertoire courant. Si ce dernier est trouvé, une valeur de -1 (vrai) sera renvoyée ou 0 (faux) le cas contraire. La fonction EXIST est capable de rechercher l'existence d'un ou de tous les fichiers du disque. Voici quelques exemples:

```
Print Exist("HAHA est vraiment un nom bête")
Print Exist("AMOS:") : Rem Est un disque appelé AMOS: disponible
Print Exist("Df1:") : Rem Est-ce-qu'un deuxième lecteur a été branché?
```

Comme vous pouvez le constater, EXIST acceptera volontiers n'importe quel charabia sans produire un seul message d'erreur. A condition que le nom du fichier comprenne au moins un caractère, EXIST poursuivra quand même sa recherche. Cependant, les chaînes vides comme "" doivent être testées séparément. Exemple:

```
F$=Fsel$("*.IFF","","Charger un fichier IFF")
IF F$="" Then Edit : Rem Revenir sur l'éditeur si aucun fichier a été choisi
If Exist(F$) Then Load Iff F$,0
```

=DIR FIRST$ *(Saisir le premier fichier du répertoire satisfaisant le nom du chemin)*

fichier$=DIR FIRST$(chemin$)
Cette fonction renvoie une chaîne contenant le nom et la longueur du premier fichier du disque qui satisfait le chemin (chemin$) de recherche courant. A l'appel de cette fonction, tout le listage du répertoire sera chargé en mémoire. Vous pouvez alors retrouver le nom du fichier suivant du répertoire en faisant un appel à la fonction DIR NEXT$.

```
Print Dir First$("*.*")
```

=DIR NEXT$ *(Saisir le fichier suivant satisfaisant le chemin courant)*

fichier$=DIR NEXT$

Cette fonction renvoie le nom du fichier suivant du listage du répertoire crée par l'appel précédent de la commande DIR FIRST$. Après que la dernière donnée de cette liste ait été lue, une chaîne sera renvoyée contenant la chaîne vide "". Le tableau du directoire sera entièrement effacé et la mémoire qu'il a consommée sera libérée et mise à la disposition du reste de votre programme Basic. Voici un exemple qui affiche tous les fichiers du répertoire courant:

```
F$=Dir First$("*.*") : Rem Lire le répertoire et saisir la première donnée
While F$<>""
   Print F$ : Bell : Wait 30
   F$=Dir Next$ : Rem Saisir le fichier suivant de la liste
Wend
```

Vous trouverez une autre démonstration de ces commandes dans le fichier EXEMPLE 20.1 du dossier du manuel. Ce programme exemple copie le contenu d'un dossier d'un disque sur un autre.

Les fichiers du disque

Les fichiers sont des collectes d'informations qui ont été groupées dans un seul endroit du disque. Chaque fichier est affecté de son propre nom pouvant comprendre 1 à 255 caractères.

Avant de pouvoir utiliser un de ces fichiers, vous devez d'abord l'initialiser au moyen des instructions OPEN IN, OPEN OUT ou de l'instruction APPEND. Lorsque vous ouvrez un fichier, vous l'affectez d'un numéro de *canal* compris entre et dix. Ce numéro sera utilisé dans toutes les opérations ultérieures sur disque pour identifier le fichier avec lequel vous êtes en train de travailler.

Le Commodore Amiga supporte deux différents types de fichiers-disque: les fichiers séquentiels et les fichiers à accès direct.

Les fichiers séquentiels

Les fichiers séquentiels sont les fichiers normalement utilisés sur l'Amiga. Ils sont appelés ainsi parce qu'ils vous permettent seulement de lire vos informations dans

l'ordre précis dans lequel elles ont été initialement créées.
Par conséquent, si vouliez changer un seul élément des données se trouvant au milieu d'un fichier séquentiel, il vous faudrait lire tout le fichier jusqu'à ce que vous rencontriez cette valeur et ensuite réenregistrer tout le fichier sur le disque.

AMOS Basic vous permet d'accéder aux fichiers séquentiels dans un but d'écriture ou de lecture mais jamais dans ces deux buts à la fois.

```
Open Out 1,"fichier.seq"
Input "Quel est votre nom";N$
Print #1,N$
Close 1
```

Cette instruction crée un fichier appelé FICHIER.SEQ contenant votre nom. Afin de relire ces informations du fichier, tapez les lignes suivantes:

```
Open in 1,"fichier.seq"
Input #1,N$
Print "Je me rappelle de ton nom. C'est";N$
Close 1
```

Remarquez la façon dont ces programmes effectuent ces trois opérations.

- Ouvrez le fichier au moyen de OPEN IN, OPEN OUT ou APPEND
- Accédez au fichier au moyen de INPUT# ou PRINT#
- Fermez le fichier au moyen de CLOSE. Notez que si vous oubliez de le fermer, les changements effectués sur ce fichier seront perdus!

Ces trois étapes doivent être suivies exactement dans cet ordre chaque fois que vous accédez à un fichier séquentiel.

OPEN OUT *(Ouvrir un fichier sortie)*

OPEN OUT canal,n$

Cette commande ouvre un fichier séquentiel d'écriture. Si ce fichier existe déjà, il sera effacé. canal représente un numéro compris entre 1 et 10 et il est utilisé pour identifier votre nouveau fichier lors de l'appel ultérieur de commandes PRINT#.

n$ est le nom du fichier à ouvrir.

APPEND *(Ajouter certaines informations à la fin d'un fichier existant)*

APPEND canal,nom$

Cette commande ouvre un fichier séquentiel de sortie. Si ce fichier existe déjà, les nouvelles données seront ajoutées à la fin de ce dernier. Cela vous permet de prolonger vos fichiers à tout moment, une fois qu'ils ont été définis.

OPEN IN *(Ouvrir un fichier d'entrée)*

OPEN IN canal,f$

Cette commande prépare un fichier à sa lecture. Si ce fichier n'existe pas, il sera automatiquement crée.

canal représente un numéro compris entre 1 et 10 qui est utilisé par les diverses instructions INPUT et qui se rapporte à votre fichier ouvert.

CLOSE *(Fermer un fichier)*

CLOSE n

Cette commande ferme le fichier numéro *n*. **Attention!** Si vous oubliez de fermer un fichier après avoir fini de l'utiliser, tous les changements que vous avez effectués ne seront pas pris en compte.

PRINT # *(Sortir une liste de variables sur un fichier ou sur une unité périphérique)*

PRINT#canal,liste de variables

Cette commande est identique à l'instruction normale print, mais au lieu d'afficher les informations à l'écran, elle les sort sur un fichier ou sur une unité périphérique de sortie spécifiée par le numéro de cannal. Voici un exemple:

```
Open Out 1,"Fichier Test"
Print#1,"Bonjour"
Close 1
```

Comme avec Print, vous pouvez abréger PRINT# par ?#.

INPUT # *(Entrer une liste de variables d'un fichier ou unité périphérique)*

INPUT #canal,liste de variables

Cette commande lit les informations d'un fichier séquentiel ou d'une unité périphérique telle qu'un port série. Comme la commande INPUT normale, elle saisit une liste de valeurs et les charge dans un ensemble de variables Basic. Comme de coutume, chaque valeur de la liste doit être séparée par une seule virgule. En outre, chaque ligne de données doit également être suivie de son propre caractère de <changement de ligne>. Celui-ci est équivalent au *Retour* sur lequel vous appuyez pour entrer une ligne depuis le clavier. Exemple:

```
Open in 1,"Fichier Test" : Rem Ouvre Fichier Créé dans Exemple Précédent
Input#1,A$
Close 1
```

LINE INPUT # (Entrer une liste de variables non séparées par une virgule)

LINE INPUT # a deux syntaxes possibles:

LINE INPUT #canal,liste de variables

ou

LINE INPUT #canal,séparateurs,liste de variables

Cette fonction est identique à INPUT#, sauf qu'elle vous permet de séparer votre liste de données en utilisant n'importe quel autre signe que la virgule. Si le caractère de séparation est omis, le système choisit automatiquement le caractère du *Retour*.

Si vous lisez un texte, choisissez toujours LINE INPUT# parce que les virgules que l'on trouve dans un texte français sont considérées comme un caractère de séparation par la commande INPUT#, ce qui dérouterait complètement votre programme.

SET INPUT *(Définir les caractères de fin de ligne)*

SET INPUT c1,c2

Cette commande définit les caractères de fin de ligne qui seront ajoutés à une ligne de données. L'Amiga suppose qu'un seul caractère de changement de ligne existe à la fin de chaque ligne alors que la plupart des autres ordinateurs (y compris le ST) veulent un Retour et un changement de ligne. Par conséquent, si vous essayez d'importer vos fichiers ST par le câble série, des douzaines de caractères *Retour* parasites apparaîtront dans vos fichiers. Vous pouvez heureusement contourner ce problème en utilisant SET INPUT.

c1 et *c2* représentent un couple de valeurs ASCII qui seront utilisées par vos caractères de séparation. Si vous voulez utiliser un seul caractère, il vous suffit d'attribuer une valeur négative à *c2* comme un moins un. Voici deux exemples:

```
Set Input 10,-1 : Rem Format Standard Amiga
Set Input 13,10 : Rem Format St
```

=INPUT$ *(Entre plusieurs caractères depuis une unité périphérique)*

x$=INPUT$ *(f,nombre)*

Cette fonction lit le nombre de caractères de l'unité périphérique ou fichier numéro *f*.

=EOF *(Test de fin de fichier)*

indicateur=EOF(canal)

EOF est une fonction utile d'AMOS Basic qui teste si la fin d'un fichier a atteint la

position courante de lecture. EOF renvoie alors une valeur de -1 ou une valeur de 0 dans le cas contraire.

LOF *(Longueur du fichier ouvert)*

long=LOF(canal)

Cette fonction renvoie la longueur d'un fichier ouvert. Il est logique d'utiliser cette fonction avec les unités périphériques autres que le disque.

POF *(Variable contenant la position courante de l'indicateur de fichier)*

pos=POF(canal)

La fonction POF change la position d'écriture ou de lecture courante d'un fichier ouvert. Par exemple:

Pof(1)=1000

Elle définit la position de lecture/écriture sur 1000 caractères, en se basant sur le début du fichier. Chose curieuse, vous pouvez vous servir de POF pour obtenir une forme rudimentaire d'accès direct lorsque vous utilisez les fichiers séquentiels! Ceci s'explique par le fait que les lecteurs de disquette sont sélectifs de par leur propre nature et par conséquent, toutes les exécutions séquentielles sont en fait simulées en utilisant l'accès direct.

Les fichiers à accès direct

Les fichiers à accès direct sont appelés ainsi parce que vous pouvez accéder aux informations contenues sur le disque dans l'ordre que vous voulez. Pour utiliser ces fichiers, vous devez d'abord avoir quelques connaissances théoriques.

Tous les fichiers à accès direct sont composés d'unités appelées enregistrements, eux même dotés de leur propre numéro. Ces enregistrements sont à leur tour divisés en un certain nombre de zones particulières. Chaque zone contient une information précise. Si vous utilisez des fichiers séquentiels, ces zones peuvent être de la longueur que vous voulez puisque le fichier sera lu dans une seule direction. Cependant, les fichiers à accès direct vous demandent de spécifier préalablement la taille maximale de chacune de ces zones.

Supposez que vous vouliez produire un fichier contenant une liste de noms et de numéros de téléphone. Dans ce cas, vous pourriez utiliser les zones de la façon suivante:

Zone	Longueur maximale
PRENOM$	15
NOM$	15
CODE$	10
TEL$	10

Vous pourriez donc définir ces zones au moyen de la ligne suivante:

```
Field 1,15 as PRENOM$,15 as NAME$,10 as CODE$,10 as TEL$
```

Il est important de bien comprendre que les chaînes spécifiées par l'instruction FIELD peuvent être également utilisées comme des variables en chaîne normales. Cela vous permet de lire et écrire des informations sur n'importe quelle zone. Par exemple:

```
PRENOM$="HILL" : Rem Charge le prénom dans la zone PRENOM$
TEST$=PRENON$ : Print Test$
```

Après avoir chargé votre enregistrement de données, vous pouvez l'écrire sur le disque au moyen de la commande PUT. Exemple:

```
Put 1,10
```

Charge les données dans l'enregistrement 10 du fichier ouvert sur le canal 1.
Similairement, vous pouvez lire un enregistrement au moyen de l'instruction GET.

```
Get 1,10
```

OPEN RANDOM (Ouvrir un canal pour le fichier à accès direct)

OPEN RANDOM canal,n$

Cette instruction ouvre un fichier à accès direct appelé n$ sur le disque courant. Si vous utilisez cette instruction, vous devez toujours définir la structure d'enregistrement immédiatement après avoir fait appel à la commande FIELD$.

FIELD *(Définir la structure d'enregistrement)*

FIELD canal, long1 AS champ1$, long2 AS champ2$.....

L'instruction FIELD vous permet de définir un enregistrement qui sera utilisé pour un fichier à accès direct. Cet enregistrement peut avoir une longueur de 65535 octets.

```
Champ 1,15 as PRENON$,15 as NAME$,10 as CODE$,10 as TEL$
```

PUT *(Placer un enregistrement dans un fichier à accès direct)*

PUT canal,r

PUT déplace un enregistrement de la mémoire de l'Amiga vers un enregistrement numéro *r* d'un fichier à accès direct. Avant d'être utilisé, le contenu du nouvel enregistrement doit d'abord être placé dans les chaînes zone définies par FIELD au moyen d'une instruction comme suit:

```
PRENOM$="HUNTINGTON"
```

Bien qu'il vous soit possible d'écrire les enregistrements existants dans l'ordre que vous voulez, vous n'êtes pas autorisé à éclater les enregistrements au hasard sur le disque. Si vous venez de créer un fichier, vous ne pouvez pas y entrer les instructions suivantes par exemple:

```
Put 1,1
Put 1,5
```

Dans ce cas, l'instruction PUT 1,5 produira un message d'erreur, puisqu'il n'existe aucun enregistrement dans le fichier dont les numéros sont compris entre 2 et 5. L'enregistrement 2 doit être le nouvel enregistrement du fichier, puis 3, 4... Exemple:

```
Open Random 1,"TELEPHONE"
Field 1,30 As NOM$, 30 As TEL$
INDEX=1
Do
   Input "Entrer un nom"; NOM$
   Exit If NOM$=""
   Input "Entrer un numéro de téléphone"; TEL$
   Put 1,Index : Inc INDEX
Loop
Close 1
```

GET *(Saisir un enregistrement d'un fichier à accès direct)*

GET canal,r

GET lit un enregistrement numéro *r* conservé dans un fichier à accès direct ouvert au moyen de la commande OPEN. Elle charge ensuite cet enregistrement dans les chaînes zone créées par FIELD. Ces chaînes peuvent être alors manipulées normalement.

Notez que vous pouvez seulement vous servir de GET pour retrouver les enregistrements qui sont réellement sur le disque. Si vous essayez de saisir un numéro d'enregistrement qui n'existe pas, un message d'erreur sera produit. Exemple:

```
Open Random 1,"TELEPHONE"
Field 1,30 As NOM$, 30 As TEL$
Do
   Input "Entrer un numéro d'enregistrement"; INDEX
   Exit if INDEX=0
   Get 1,INDEX : Print NOM$ : Print TEL$
   Loop
Close 1
```

L'imprimante

Si vous possédez une imprimante, vous pourrez sortir vos programmes AMOS. C'est une aide précieuse pendant le débogage de vos programmes et elle doit être considérée comme un achat indispensable pour les programmateurs sérieux. Les commandes suivantes vous donnent accès à l'imprimante:

LPRINT *(Sortir un liste de variables sur imprimante)*

LPRINT liste de variables

Cette commande est identique à PRINT mais elle envoie vos données vers l'imprimante au lieu de les afficher à l'écran. Exemple:

```
Lprint "Bonjour"
```

Voir PRINT, USING, PRINT#

LDIR *(Sortir un répertoire sur imprimante)*

LDIR [chemin$] [/W]

Cette commande permet d'imprimer le répertoire du disque courant. Voir DIR pour plus de précisions.

Les unités périphériques externes

OPEN PORT *(Ouvrir un canal vers un port E/S)*

OPEN PORT channel,"PAR:" (Ouvre un canal vers l'interface parallèle)
OPEN PORT channel,"SER:" (Ouvre un canal vers le port RS232)
OPEN PORT channel,"PRT:" (Ouvre un canal vers l'imprimante sélectionnée)

OPEN PORT vous permet de communiquer avec les unités périphériques externes telles que le port RS232. Vous pouvez exécuter toutes les commandes de fichiers séquentiels standard comme de coutume, à l'exception des commandes LOF ou POF qui s'appliquent seulement aux opérations sur disque. Exemple:

```
Open Port 1,"SER:"
For X=0 to 10
   Print#1,"AMOS BASIC"
Next X
Close 1
```

Ce programme sort dix lignes de texte sur l'unité périphérique branchée au port RS232. Si votre imprimante utilise un port parallèle, remplacez la ligne 10 par:

```
Open Port 1,"PRT:"
```

De la même manière, vous pouvez entrer des informations depuis une unité périphérique telle qu'un modem avec la ligne suivante par exemple:

```
Input#1,A$ : Print A$
```

Lorsque vous accédez à ces unités périphériques externes, vous pouvez vous servir de toutes les instructions normales d'entrée, y compris INPUT$ et LINE INPUT.

=PORT *(Tester l'attente du canal)*

n=PORT(canal)

Cette fonction teste si une unité périphérique d'entrée est prête à vous envoyer des informations. Si cette unité attend que vous les lisiez, une valeur de -1 (vrai) est renvoyée par cette fonction. Dans le cas contraire, une valeur de 0 (false) sera communiquée.

Rechercher les périphériques courants

AMOS peut vous rendre compte de la liste courante de périphériques grâce aux commandes suivantes:

=DEV FIRST$ *(Saisir le premier périphérique de la liste courante des périphériques)*

dev$=DEV FIRST$("filtre")

Cette instruction s'exécute de la même façon que Dir First$ et Dir Next$, mais renvoie la liste de périphériques. Notez que vous devez effacer les espaces avec -"" pour obtenir le nom correct.

=DEV NEXT$ *(Saisir le périphérique suivant satisfaisant le filtre)*

dev$=DEV NEXT$

Cette instruction saisit le périphérique suivant de la liste des périphériques. Une chaîne vide indique la fin de la liste. Exemple:

```
Print Dev First$
Do
   A$=Dev Next $
   A$=A$-""
   If A$="" then End
   Print A$
Loop
```

21 Compactage d'écran

Bien que la machine de l'Amiga soit capable d'afficher des images étonnantes, il a été pratiquement impossible, jusqu'à maintenant, d'obtenir ces effets dans un jeu réel. C'est simplement un problème de mémoire. Un seul écran de 64 couleurs mesurant 320 sur 255 pixels consomme une mémoire vive de 60k qui est donc très importante. Cela impose une limite réelle au nombre d'écrans que vous pouvez utiliser dans vos jeux, particulièrement si vous souhaitez que vos programmes se déroulent sur un A500 non étendu.

Ce problème est résolu en produisant des écrans à l'aide du définisseur de carte AMOS. Cette solution convient aux écrans en arrière-plan dans un jeu d'arcade, mais elle ne l'est pas vraiment pour le type d'images réelles que vous pouvez trouver dans un jeu d'aventure.

Heureusement, il existe une autre solution: ce sont les instructions de compactage AMOS. Celles-ci peuvent rapidement compacter un écran entier en une fraction de sa taille initiale. Tous les modes graphiques standard sont supportés, y compris le HAM. Il n'y a donc rien qui vous empêche d'intégrer de fantastiques images dans vos programmes AMOS.

SPACK *(Compactage d'écran)*

SPACK s TO n [tx,ty,bx,by]

La commande SPACK, (prononcée s-pack) comprime l'écran *s* dans la banque mémoire *n*. Tout ce qui se rapporte à l'image courante est sauvegardé, y compris son mode, sa taille d'écran, son décalage et sa position d'affichage. Cette commande vous permet de reproduire exactement la copie d'origine de votre écran.

s est le numéro de l'écran contenant votre image. *n* représente le numéro d'une banque mémoire compris entre 1 et 16. Si cette banque n'existe pas, elle sera réservée automatiquement pour vous. Votre nouvelle banque sera conservée en mémoire FAST si elle est disponible et sera sauvegardée avec votre programme Basic. Après avoir appelé cette fonction, vous pourrez trouver la taille de votre écran à l'aide de LENGTH. Exemple:

```
f$=FSEL$("*.Iff","","Charger Une Image")
Load Iff F$,0
Spack 0 To 1:Rem Compacter Ecran Zero En Un Fichier De La Banque Une.
Print "La Longueur De Votre Nouvelle Banque Est ";Length(1);" Octets"
Wait Key
Screen Close 0
Unpack 1 To 0:Rem Reproduire Ecran Compacté
```

Bien évidemment, vous n'avez pas à compacter un écran entier avec cette instruction.

Les paramètres optionnels vous permettent de comprimer n'importe quelle portion rectangulaire de l'image que vous voulez.

tx,ty représentent les coordonnées de l'angle supérieur gauche de cette zone. *bx,by* déterminent la position de l'angle inférieur droit de votre image. Toutes les abscisses sont arrondies à la frontière la plus proche de 8 pixels.

Notez que pour obtenir la réduction maximale de la mémoire, SPACK essaiera de compacter votre image en utilisant plusieurs stratégies. Elle comprimera votre image au moyen de la méthode qui consomme le moins de mémoire. En conséquence, la compression d'une image se fait en 6 secondes environ. Ce n'est guère un inconvénient puisque le compactage est seulement nécessaire lors de la rédaction de vos programmes.

Puisque chaque image peut être décondensée en moins d'une seconde, vous n'avez pas à craindre des interférences avec la vitesse de vos jeux. Si la vitesse est très importante, utilisez plutôt le système CBLOCK. Voir le chapitre sur les graphiques en arrière-plan pour plus de précisions.

Si, par hasard, vous comparez la taille compactée de vos fichiers à leur longueur d'origine sur le disque, il se peut que vous soyez induit en erreur en sous-estimant la réduction de la taille de la mémoire. Il est important de bien comprendre que la plupart de ces fichiers ont DEJA ETE COMPRIMES au moyen de routines standard de compactage IFF. Il est donc plutôt surprenant qu'AMOS puisse les réduire encore de 20%!

Les écrans compactés sont parfaits pour les titres et les tableaux de marquage des points d'un jeu d'arcade puisqu'ils vous permettent d'introduire des effets chouettes sans consommer beaucoup de mémoire. Ils peuvent être également intégrés directement dans les GAP et les jeu d'aventures.

PACK (Comprimer un écran)

PACK s TO n [tx,ty,bx,by]

Cette commande comprime un écran *s* en une banque numéro *n*. A la différence de la commande précédente SPACK, seules les données de l'image sont condensées. Par conséquent, votre écran compacté doit toujours être décomprimé directement dans un écran existant.

En raison de la façon dont les images sont décomprimées, vous apercevrez un effet miroitant sauf si vous avez préalablement double tamponné vos écrans. Essayez d'éviter l'usage de PACK avec les écrans à un seul tamponnement. Il est bien plus sensé d'appeler le système SPACK à cette fin.

Si les coordonnées optionnelles sont précisées, seule une portion de l'image sera comprimée.

tx,ty spécifient la position de l'angle supérieur gauche de cette zone. *bx,by* représentent les coordonnées de l'angle inférieur de votre image. Comme auparavant, toutes les abscisses sont arrondies à la frontière la plus proche de 8 pixels.

Puisque PACK est tout à fait compatible au sytème standard Autoback, il est facile de combiner les images compactées et les écrans mobiles. Si vous utilisez le mode Autoback2, vous pourrez même décomprimer vos images DERRIERE les bobs

existants. Il est donc possible d'utiliser cette instruction avec SCREEN OFFSET pour créer de fantastiques scrollings en arrière-plan dans vos jeux.

UNPACK *(Décomprimer un écran compacté)*

Cette instruction décomprime un écran qui a été préalablement compacté au moyen des instructions SPACK ou PACK. Chaque routine de compactage a sa propre version de la commande UNPACK.

SPACK

UNPACK b TO s

Cette instruction ouvre un écran *s* et remet l'écran compacté dans la banque *b*. Si cet écran existe déjà, il sera remplacé par la nouvelle image. L'écran décomprimé sera proprement visualisé.

PACK

Les écrans comprimés peuvent être décomprimés au moyen de deux instructions. Celles-ci saisissent une image d'une banque mémoire et la charge directement dans un écran existant. **Attention!** L'écran de destination **doit** être exactement du même format que votre image compactée, sinon vous obtiendrez un message d'erreur *appel de fonction interdit.*

Lorsque vous décomprimez vos écrans, vous remarquerez peut-être que l'effet produit est légèrement désordonné. La commande PACK est seulement destinée à être utilisée par le système à double tamponnement. Vous obtiendrez un effet régulier et ravissant à condition que votre écran soit double tamponné.

UNPACK b

Cette instruction décomprime l'écran à sa position d'origine.

UNPACK b,x,y

Cette instruction retrace votre image à partir des coordonnées *x,y*. Si la nouvelle image n'entre pas dans l'écran courant, vous obtiendrez un message d'erreur.

22 Code machine

AMOS Basic comprend un jeu de commandes permettant au programmateur confirmé d'avoir accès au fonctionnement interne de la machine de l'Amiga. Aucune de ces instructions ne sont bien sûr nécessaires à la programmation générale.

AMOS Basic a été soigneusement conçu pour vous permettre de contrôler toutes les fonctions de l'Amiga en utilisant des instructions Basic simples et faciles à comprendre. Vous n'avez donc pas vraiment besoin de solliciter directement la mémoire pour créer vos jeux!

Conversion d'un nombre

=HEX$ (Convertir un nombre en un caractère hexadécimal)

h$=HEX$(v)
h$=HEX$(v,n)

Cette fonction convertit le nombre entier *v* en une notation hexadécimale (base 16). Elle renvoie une séquence de caractères hexadécimaux dans la chaîne *h$*. Exemples:

```
Print Hex$(Colour(1),3)
$A40
Print Hex$(65536)
$10000
Print Hex$(65536,8)
$00010000
```

=BIN$ *(Convertir un nombre en une chaîne binaire)*

b$=BIN$(v)
b$=BIN$(v,n)

Cette fonction convertit un nombre en une notation binaire (base 2). Comme avec HEX$, vous pouvez choisir de sortir tous les chiffres d'un nombre ou seulement quelques-uns. Exemple:

```
Print Bin$(255)
%11111111
Print Bin$(255,16)
%0000000011111111
```

n spécifie le nombre de chiffres binaires qui seront renvoyés dans *b$*. Chaque nombre peut comprendre 1 à 32 chiffres (1<=n<=31).

Manipulation de la mémoire

=PEEK *(Saisir l'octet à une adresse)*

v=PEEK(adresse)

Cette fonction renvoie l'octet de 8 bits rangé à une adresse).

```
Load "AMOS_DATA:Sprites/Monkey_right.abk"
Print Peek(Start(1))
For S=8 To 1 Step -1
   C=peek(start(1)-S) : Print CHR$(c); : Rem C'est donc là que se trouve le nom
Next S
```

POKE *(Changer l'octet à une adresse)*

POKE adresse,v

Cette fonction copie le nombre v dans une adresse. v doit toujours représenter une valeur comprise entre 0 et 255.

Vous pouvez utiliser cette fonction pour changer le contenu de n'importe quelle partie de la mémoire de l'Amiga. Mais sachez que POKE peut être très dangereux. Si vous écrivez à tout hasard dans la mémoire vive, vous allez bloquer votre Amiga. Exemple:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Poke (Start(1)-5),asc("o") : Rem Changer une seule lettre du nom de la banque
Listbank
```

=DEEK *(Lire le mode à une adresse)*

v=DEEK(adresse)

Cette fonction lit le mot à deux octets trouvé à *une adresse. L'adresse* DOIT avoir une valeur paire sinon vous obtiendrez un message d'erreur d'adresse.

DOKE *(Changer le mot à une adresse)*

DOKE adresse,valeur

Cette fonction charge un nombre à deux octets compris entre 0 et 65535 dans l'emplacement de la mémoire à une *adresse*. Cette fonction peut être utile.aux connaisseurs. Puisque l'erreur est humaine, vous devriez toujours sauvegarder une copie de vos programmes sur disquette avant d'essayer d'utiliser cette fonction dans une nouvelle routine. Exemple:

```
Doke Phybase(1)+1000,65535
```

=LEEK *(Lire un long mot à une adresse)*

v=LEEK(adresse)

Cette fonction renvoie un mot de quatre octets rangé à *une adresse*. Comme DEEK, l'adresse utilisée avec cette fonction doit toujours avoir une valeur PAIRE. Si le bit 31 de la valeur renvoyée est forcé à 1, *v* sera affiché d'une valeur négative. Ce n'est pas une erreur. C'est seulement un effet résultant de la façon dont AMOS traite les nombres. Exemple:

```
Print Leek(0)
```

LOKE *(Changer un long mot à une adresse)*

LOKE adresse,n

LOKE copie le nombre *n* à quatre octets dans l'adresse. Exemple:

```
Loke phybase(1)+10000,$FFFFFFFF
```

L'utilisation imprudente de cette fonction peut amener l'Amiga à se bloquer désastreusement; faites donc attention.

=VARPTR *(Saisir l'adresse d'une variable)*

adresse=VARPTR(variable)

Cette fonction renvoie l'adresse dans la mémoire d'une variable Basic. Chaque type de variable est rangé dans son propre format:

Nombres entiers: VARPTR trouve l'adresse des quatre octets renfermant le contenu de votre variable. Exemple:

```
A=0
Loke Varptr(A),1000
Print A
1000
```

Virgule flottante: VARPTR renvoie l'emplacement des quatre octets représentant la valeur de la variable dans le seul format de précision du bus GPIB.
Chaînes: L'adresse VARPTR désigne le premier caractère de la chaîne. Puisque les chaînes d'AMOS Basic ne sont pas terminées par un CHR$(0), il vous faut obtenir la longueur de la chaîne au moyen d'une instruction comme DEEK(VARPTR(A$)-2), où A$ est le nom de votre variable. Bien sûr, vous pouvez également utiliser LEN(A$).

COPY *(Copier un bloc de mémoire)*

COPY début,fin TO destination

Cette commande est utilisée pour déplacer rapidement de grands blocs de mémoire d'un endroit à l'autre. *début* et *fin* sont respectivement les adresses du premier et du dernier octet de vos données. *destination* désigne la zone de mémoire qui sera chargée de vos nouvelles données.

Toutes ces adresses **DOIVENT** avoir une valeur paire sinon vous obtiendrez un message d'erreur d'adresse. Exemple:

```
Reserve As Data 10,000
A$="Salut"
Copy Varptr(A$),Varptr(A$)+Len(A$)+1 To Start(10)
For P=0 To Len(A$)
   Print Chr$(Peek(Start(10)+P));
Next P
```

FILL *(Garnir un bloc de mémoire d'un long mot)*

FILL début TO fin,motif

Cette commande garnit une zone sélectionnée de la mémoire avec les quatre octets contenus dans *motif*.

début et fin déterminent la postion et la taille du bloc à garnir. **Attention!** Ces adresses DOIVENT avoir une valeur paire!

motif est un long mot contenant un motif de remplissage de quatre octets. Il sera copié dans chaque groupe de quatre adresses de mémoire entre *début* et *fin*. Exemple:

```
Reserve As Data 10,1000
Rem Créer une chaîne de quatre caractères (4 octets=1 Longmot)
A$="AMOS"
Rem Garnir la banque d'une chaîne
Fill Start(10) To Start(10)+Length(10),Leek(Varptr(A$))
For P=0 To Length(10)
   Print Chr$(Peek(Start(10)+P));
Next P
```

=HUNT *(Trouver une chaîne en mémoire)*

f=HUNT (début TO fin, s$)

Cette fonction recherche la séquence de caractères contenue dans *s$*. *début* est l'adresse du premier octet en mémoire à rechercher et *fin* est l'adresse du dernier octet.

Lorsque la recherche est terminée, *f* contient la valeur 0 si la chaîne dans a$ n'a pas été trouvée ou l'adresse de *f$* dans le cas contraire.

Opérations en mode binaire

ROL *(Faire une permutation circulaire vers la gauche)*

ROL.B n,v
ROL.W n,v
ROL.L n,v

ROL est une version Basic de l'intruction ROL que l'on retrouve dans le langage assembleur 68000. Prenez la représentation binaire d'un nombre en *v* et permutez-le dans exactement n positions à partir de la gauche.

Si *v* est une seule variable, alors le nombre à permuter est pris directement de *v*. Mais si *v* est une expression, elle est alors considérée comme **l'adresse** de votre nombre. Voici un exemple:

Le nombre 136 est représentée en binaire par:

%10001000

Entrez:

```
V=136
Rol.B 1,V
Print Bin$(V,8)
%000 10001
```

Constatez que le premier bit du nombre est réapparu magiquement de derrière. Voici un exemple du système d'adresse:

```
Reserve As Work 10,1000
Poke Start(10),%00001111
Rol.B 1,Start(10)
Print Bin$(Peek(Start(10)),8)
%00011110
```

Cette instruction donne le nombre 17 ou son équivalent binaire %00010001.

Comme vous voyez, le nombre entier a été déplacé vers la gauche, et le 1 le plus haut a été ramené à la position la plus basse. Le ".B" se trouvant après Rol permet d'instruire AMOS de considérer ce nombre comme un octet à 8 bits. Vous pouvez également spécifier les tailles ".W" (mot) et ".L" (long mot). Exemples:

```
Rem Noter le . entre Rol et L
A=1
Rol.L 1,A
Print A
2
```

ROL peut être utilisé pour multiplier très rapidement un nombre positif par une puissance de 2.

ROR *(Faire une permutation circulaire vers la droite)*

ROR.B n,v
ROR.W n.v
ROR.L n.v

Cette instruction est très similaire à ROL mais permute un nombre en sens inverse. Comme auparavant, *v* peut être une simple variable ou une expression. Si c'est une expression, ROL permutera le nombre rangé à **l'adresse résultante**. Exemple:

```
A=8
Ror 1,A
Print A
4
```

ROR est capable de diviser n'importe quelle valeur positive par une puissance de deux. Le calcul résultant sera effectué bien plus rapide que l'équivalente opération "/".

=BTST *(Tester un bit)*

b=BTST(n,v)

Cette fonction teste le chiffre binaire à la position *n* de la variable *v*. Si *v* est une expression, elle sera utilisée comme l'adresse du bit qui doit être recherchée. Dans ce cas, une intersection logique sera automatiquement établie entre *n* et 7 avant l'exécution de la fonction.

Après que BTST ait été appelé, *b* est attribué de la valeur -1 (vrai) si le bit à la position *n* est forcé à 1, sinon il aura la valeur 0 (faux). Exemple:

```
B=%1010
Print Btst (3,B)
-1
Print Btst (2,B)
0
```

Voir également BCHG, BCLR, BSET

BSET *(Forcer un bit à 1)*

BSET n,v

Cette instruction force à 1 le bit à la position *n* dans la variable *v*. Si vous remplacez une expression par cette variable, elle sera considérée comme l'adresse d'une valeur dans

la mémoire de l'Amiga. Exemple:

```
A=0
Bset 8,A
Print A
256
```

BCHG *(Changer un bit)*

BCHG n,v

Cette instruction change le bit numéro *n* de la variable *v*. Si ce bit est de 1, la nouvelle valeur sera alors de 0 et vice versa. Comme de coutume, si *v* est remplacé par une expression, le résultat sera considéré comme une adresse. Exemple:

```
A=0
Bchg 1,A
Print A
2

Bchg 1,A
Print A
0
```

BCLR (Supprimer un bit)

BCLR n,v

Cette instruction supprime le bit numéro *n* de la variable *v* en le forçant à zéro. Comme toutes les opérations en mode binaire, si *v* est une expression, elle sera utilisée en tant qu'adresse de vos données en mémoire. Exemple:

```
A=128
Bclr 7,A
Print A
0
```

Comment utiliser le langage assembleur

AMOS Basic comprend des fonctions spéciales vous permettant d'intégrer les routines du langage assembleur à vos programmes Basic. Il serait bon de souligner que le code machine est rarement utile en raison de la puissance réelle du système AMOS. Nous avons seulement ajouté ces fonctions pour les programmeurs en langage assembleur qui peuvent souhaiter optimiser leurs programmes Basic avec un petit peu de code machine.

Attention! Ces commandes sont extrêmement dangereuses et vous pouvez bloquer

votre Amiga si vous les utilisez imprudemment. A moins que vous soyez tout à fait à l'aise avec la complexité du hardware de l'Amiga, nous vous conseillons d'éviter l'usage de ces fonctions!

PLOAD *(Réserver une banque mémoire pour le code machine)*

PLOAD "nomfichier",banque

Cette commande réserve une banque mémoire et la charge d'un programme en code machine depuis le disque.

banque est le numéro de la banque mémoire devant être réservée pour votre programme. S'il est négatif, alors la banque sera calculée en utilisant la valeur absolue de ce nombre et l'affectation de la zone de mémoire nécessaire se fera depuis la mémoire de la PUCE.

Vous pouvez sauvegarder ce programme sur disque comme un fichier ".ABK" normal. Puisque les banques créées par cette fonction sont permanentes, elles seront également sauvegardées directement avec vos programmes AMOS.

Votre programme doit être un fichier code machine de forme standard Amiga. Il peut contenir pratiquement tout ce que vous voulez mais il est limité par les restrictions suivantes:

- Le code **doit** être relogeable parce qu'il sera positionné à la première adresse de mémoire libre.
- Seule la tranche de **code** de votre programme sera chargée.
- Le programme doit être terminé par une seule instruction RTS.

CALL *(Appeler un programme en code machine)*

CALL adresse[,params]
CALL banque[,params]

Cette commande exécute un programme en langage assembleur contenu dans la mémoire de l'Amiga.

adresse peut être l'adresse absolue de votre code ou le numéro d'une banque mémoire AMOS qui a été préalablement créée à l'aide de la commande PLOAD.

Au lancement de votre programme, les registres de D0 à D7 et de A0 à A6 sont chargés des valeurs contenues dans les tableaux DREG et AREG. Votre programme en langage assembleur peut changer n'importe quel des 68000 registres. A la mise en route de la routine, le registre A3 désigne la liste optionnelle de paramètres et A5 contient l'adresse de la principale zone de données AMOS. Lorsque l'exécution de votre programme est terminée, vous pouvez revenir au Basic au moyen d'une simple instruction RTS.

params est une liste de paramètres qui sera empilée sur la pile d'A3 par la commande CALL. Ces paramètres doivent être déplacées en ordre INVERSE. Par conséquent, la dernière valeur que vous avez entrée dans l'instruction sera la première de la pile. Selon le type de vos paramètres, les valeurs référencées par A3 auront l'une des trois formes suivantes:

Nombre entier: Il représente un long mot contenant un nombre entier normal AMOS.

Virgule flottante: Contient un nombre à virgule flottante dans le seul format de précision du bus GPIB.

Chaîne: Conserve l'adresse de la chaîne. Toutes les chaînes commencent par un mot représentant leur longueur.

Attention! N'entrez jamais une chaîne directement en mémoire vive! Lorsqu'une chaîne est initialisée à une constante, l'adresse de la chaîne désigne l'instruction d'affectation d'origine dans le listing courant du programme! Si vous changez cette valeur, vous modifiez le code source d'origine. C'est évidemment très imprudent et à éviter.

Pour une démonstration de PLOAD et de CALL, référez-vous au fichier **EXEMPLE 21.1** du dossier MANUEL.

=AREG *(Variable utilisée pour passer des informations aux registres d'adresse du processeur 68000)*

a=AREG(r)
AREG(r)=a

Cette fonction est un tableau de six PSEUDO variables qui sont utilisées pour contenir une copie des six premiers registres d'adresse du processeur 68000. *r* peut être compris entre 0 et 6 et indique le numéro du registre d'adresses qui doit être affecté.

Lors de l'exécution de la commande CALL, le contenu de ce tableau se charge automatiquement dans les registres d'adresse de A0 à A6; après l'exécution de cette fonction, il est sauvegardé avec toute nouvelle information qui a été placée dans les registres correspondants.

=DREG *(Variable utilisée pour passer des informations aux registres de données du processeur 68000)*

d=DREG(r)
DREG(r)=d

C'est un tableau de huit nombres entiers qui contient une copie du contenu des 68000 registres de données. *r* se rapporte au numéro de registre et il peut être compris entre 0 et 7 correspondant aux registres D1 à D7 respectivement.

Comment accéder aux bibliothèques du système

AMOS vous permet également d'appeler la plupart des bibliothèques du système interne de l'Amiga directement depuis la mémoire morte. Ces bibliothèques ne sont pas particulièrement utiles puisque tous les appels vraiment intéressants ont déjà été incorporés dans l'AMOS!

N'utilisez pas ces routines à moins que vous sachiez parfaitement ce que vous faites. L'Amiga est bien connu pour être difficile à programmer et il est très facile de bloquer le système et de produire le détestable message d'erreur GURU par inadvertance.

=DOSCALL *(Bibliothèque DOS)*

r=DOSCALL(fonction)

DOSCALL exécute une fontion directement depuis la bibliothèque DOS. *fonction* représente le déplacement à la fonction appropriée. Référez- voûs aux Manuels Kernel de la mémoire morte de l'Amiga pour plus de précisions.

Avant d'utiliser cette fonction, il vous faudra définir quelques-uns des compteurs d'instructions de D0 à D7 et de A0 à A6 au moyen des fonctions AREG et DREG. Lorsque la routine revient au Basic, le contenu de D0 est renvoyé dans *r*. **Note:** Les valeurs de retour ne sont pas chargées dans DREG et AREG.

=EXECALL *(Bibliothèque EXEC)*

=EXECALL(fonction)

EXECALL effectue un appel à la bibliothèque EXEC de l'Amiga. A leur entrée, les registres de D0 à D7 et de A0 à A6 sont chargés des définitions de commande saisies dans les tableaux DREG et AREG. *r* est renvoyé renfermant le contenu de D0.

=GFXCALL *(Bibliothèque des graphiques)*

r=GFXCALL(fonction)

Cette fonction appelle une routine de la bibliothèques des graphiques en utilisant les valeurs d'intruction conservées dans les tableaux DREG et AREG.

fonction représente le déplacement à la fonction correspondante. *r* est le résultat de l'opération renvoyée dans D0.

=INTCALL (Bibliothèque Intuition)

r=INTCALL(fonction)

INTCALL exécute une commande de la bibliothèque Intuition. Comme de coutume, les valeurs d'instruction sont chargées depuis les tableaux DREG et AREG et *r* contient le résultat de l'appel.

Puisqu'AMOS n'utilise pas la routine Intuition classique, cette fonction et particulièrement dangereuse. Appelez-la seulement si vous êtes déjà familier avec la bibliothèque Intuition de l'Amiga.

A l'intérieur de l'AMOS Basic

Afin de permettre l'accès à tous les fonctionnements internes du système AMOS aux réalisateurs, nous avons prévu plusieurs points d'accueil dans les diverses zones de données. Ceux-ci ne sont pas adressés au programmateur amateur mais permettent aux utilisateurs confirmés de créer leur propres utilitaires AMOS.

=SCREEN BASE *(Saisir la table d'écran)*

table=SCREEN BASE

Cette fonction renvoie l'adresse de base de la table interne utilisée pour contenir le numéro et la position de vos écrans AMOS. Pour une simple démonstration, référez-vous au **fichier EXEMPLE 21.2.**

=SPRITE BASE *(Saisir la table de sprite)*

table=SPRITE BASE (n)

Cette fonction donne l'adresse de la liste interne de données concernant le sprite *n*. Si ce sprite n'existe pas, l'adresse de la *table* aura la valeur zéro.

Les valeurs négatives de *n* renvoient l'adresse du MASK optionnel correspondant au sprite. *table* contient une des trois valeurs possibles, en fonction de l'état du masque:

table<0 Indique qu'il n'existe aucun masque pour ce sprite.
table=0 Le sprite n n'a pas de masque mais celui-ci n'a pas encore été produit par le système.
table>0 C'est l'adresse du masque en mémoire. Le premier long mot de cette zone contient la longueur du masque et le mot suivant précède la définition réelle.

Voir le **fichier EXEMPLE 21** du dossier MANUEL.

=ICON BASE *(Saisir la table d'icône)*

table=ICON BASE(n)

Cette fonction renvoie l'adresse de l'icône *n*. La syntaxe de cette fonction est exactement la même que la précédente SPRITE BASE.

Extension du Demandeur du Système

REQUEST

REQUEST ON

Cette commande par défaut permet à AMOS de produire sa propre routine de demandeur (Requester).

REQUEST OFF

AMOS sélectionne toujours le bouton ANNULER du demandeur si cette commande est utilisée. Le demandeur en question ne sera pas affiché; c'est donc idéal pour l'interception des erreurs dans un programme.

REQUEST WB

Cette commande demande à AMOS de repasser au demandeur du système du Workbench. Il vous faudra revenir sur AMOS dès que vous avez choisi une des options.

Note: Si vous n'avez pas chargé le Demandeur (en l'annulant de la liste d'extension dans le fichier config), le demandeur normal du Workbench sera utilisé pour afficher des messages.

Cependant, ceci entraîne des effets pervers. AMOS semblera se bloquer à l'apparition d'un demandeur. Si cela se produit, appuyez simplement sur Amiga+A pour revenir sur le Workbench, répondez à la question et appuyez de nouveau sur Amiga+A pour revenir sur AMOS. C'est la meilleure façon d'éviter de charger le demandeur si la mémoire est très faible!

23 Extension de périphérique série

Bienvenue au monde fascinant des communications AMOS. L'extension série est destinée à ceux d'entre vous qui souhaitent faire un transfert d'informations entre plusieurs ordinateurs utilisant le port série d'Amiga.

L'extension vous offre 15 nouvelles commandes. Vous pouvez les utiliser pour créer de fantastiques jeux multi-utilisateurs. Il est également possible de solliciter une interface Midi qui a été branchée à un port série de l'Amiga. Les musiciens pourront donc contrôler leurs instruments directement depuis AMOS Basic. Sensass! Comme vous vous y attendez, le multitâche est entièrement supporté.

Ouvrir le port série

SERIAL OPEN *(Ouvre un canal pour le circuit sériel d'Entrée/Sortie)*

SERIAL OPEN Canal, Port_no [,Partagé, Xdésactivé, 7câbles]

Ouvre un canal de communication au périphérique série.

Canal: C'est un numéro d'identificaton qui sera utilisé pour toutes les commandes ultérieures de communication. Les valeurs autorisées sont comprises entre 0 et 3.
Port_no Spécifie le numéro du périphérique logique du port série. Normalement, cette valeur devrait être forcée à zéro. Cependant, si vous avez installé une carte MULTI SERIE dans votre Amiga, vous pouvez accéder à d'autres ports en utilisant des numéros à partir de un.

Partagé (optionnel): C'est un indicateur qui informe AMOS que le périphérique peut être partagé avec d'autres tâches qui sont en cours d'exécution sur votre Amiga. Il est utilisé en multitâche. Une valeur de FAUX (zéro) saisit le canal d'AMOS Basic et refuse l'accès à tout autre programme. S'il est forcé sur VRAI (-1), le port série peut être partagé entre plusieurs programmes en mémoire. Attention: Ce système doit être utilisé avec beaucoup d'attention ou l'Amiga pourrait facilement se bloquer!

Xdésactivé (optionnel): Bascule le contrôle XON/XOFF lors de la transmission de vos données sur la ligne sérielle. Il est important de définir cet indicateur lors de la première ouverture du périphérique, même s'il sera seulement utilisé plus tard. La valeur implicite est FAUX (0) et désactive le système. Si vous voulez activer le contrôle, utilisez une valeur de VRAI
(-1). Après avoir ouvert le port, il vous faudra mettre en place les caractères XON et XOFF en faisant un appel particulier à la commande SERIE X.

7Câbles (optionnel): Une valeur de VRAI (-1) informe le périphérique d'utiliser le système à 7 câbles comme l'explique le manuel de référence Commodore. Le défaut est FAUX (0).

Lorsque vous appelez la commande Ouvrir Série pour la première fois, elle chargera automatiquement la bibliothèque PERIPHERIQUE.SERIE (SERIAL.DEVICE) depuis la disquette système. Assurez-vous qu'elle est placée dans le lecteur courant.

Si les paramètres série implicites utilisent le protocole du Minitel français: 1200 Baud, 7 bits, 1 binaire d'arrêt, parité paire. Vous pouvez facilement les modifier en utilisant au besoin les instructions VITESSE SERIE (SERIAL SPEED), BIT SERIE (SERIAL BIT) ou PARITE SERIE (SERIAL PARITY).

Fermer le port série

SERIAL CLOSE *(Ferme un ou plusieurs canaux sériels)*

SERIAL CLOSE [Canal]

Si vous omettez le numéro de canal, SERIAL CLOSE fermera tous les canaux série couramment ouverts sans aucun contrôle d'erreurs.

Le numéro optionnel de canal vous permet de fermer un seul canal et d'utiliser tous les contrôles d'erreurs normaux.

Note: Si un programme est LANCE depuis AMOS Basic, tous les canaux ouverts seront automatiquement fermés.

Envoyer des informations par le port série

SERIAL SEND *(Sortir une chaîne par un canal sériel)*

SERIAL SEND Canal, T$

Envoie directement la chaîne *T$* au canal sériel spécifié. Elle n'attend pas que les données soient transmises par le port existant.

Il vous faudra donc utiliser la fonction =SERIAL CHECK(Canal) pour détecter la complétude de la transmission.

SERIAL OUT (Sort un bloc de mémoire par un canal sériel)

SERIAL OUT Canal, Adresse, Longueur
Elle est identique à Serial Send excepté qu'elle fonctionne avec des données BRUTES.
Adresse est l'adresse de vos données en mémoire.

Longueur est le nombre d'octets à envoyer.

Lire des informations depuis le port série

SERIAL GET *(Saisit un octet d'un périphérique série)*

=SERIAL GET (Canal)

Lit un seul octet depuis le périphérique série. Note: Si aucun octet n'est disponible, une valeur de -1 sera renvoyée.

SERIAL INPUT$ *(Saisit une chaîne du port série)*

=SERIAL INPUT$ (Canal)

Lit une chaîne entière de caractères depuis le port série. S'il n'y a pas de données, la commande renverra une chaîne vide "". La chaîne contiendra autrement tous les octets qui ont été transmis jusqu'à présent par le canal sériel.

Soyez prudent lorsque vous utilisez cette commande avec des transferts à grande vitesse (tel que MIDI). Si vous attendez trop longtemps entre l'exécution de chaque commande SERIAL INPUT$, vous pouvez surcharger le système et produire des messages d'erreurs ennuyeux tels que "chaîne trop longue" ou "débordement du tampon du périphérique série".

Définir les paramètres série

SERIAL SPEED *(Définit la vitesse de transfert de baud pour un canal sériel)*

SERIAL SPEED Canal, vitesse de transmission

Définit la vitesse courante de transfert du canal approprié. La même valeur sera utilisée pour les opérations de lecture et d'écriture. Notez que vous ne pouvez pas diviser les vitesses de transmission pour un seul canal.

Si la vitesse de transmission que vous avez spécifiée n'est pas supportée par le périphérique courant, il peut être rejeté par le système.

SERIAL BIT *(Définit les parties Nbit et Binaire d'arrêt d'un protocole)*

SERIAL BIT Canal, Nbits, Binaire d'arrêt

Affecte le nombre de bits qui seront utilisés pour chaque caractère que vous transmettez.

Nbits est le nombres de bits
Binaire d'arrêt spécifie le nombre de binaires d'ARRET

SERIAL PARITY *(Définit le contrôle de parité)*

SERIAL PARITY Canal, Parité

Définit le contrôle de parité que vous utilisez pour le canal sériel courant. Voici une liste des options disponibles.

PARITE <0: Pas de parité
PARITE >0: Seuls les deux premiers bits de cette valeur sont importants.

Bit 0: =0 -> PAIR (EVEN)
=1 -> IMPAIR (ODD)
Bit 1: =0 -> parité normale
=1 -> ESPACE

Le bit de parité peut être défini en utilisant les fonctions BSET ou BCLR de la manière suivante:

P=0 : Bset P,I : Rem parité impaire
Bclr 1,P : Rem parité normale
Parité série 1,P : Rem Définir la parité en utilisant la valeur en P

Consultez le manuel de référence Commodore qui vous donnera une explication complète de ce système.

SERIAL X *(Définit XON/XOFF)*

SERIAL X Canal, modeX

Une valeur de VRAI pour le modeX désactive l'échange de données. Toute autre valeur le réactive. Le *modeX* doit recevoir les caractères de commande corrects. Ceux-ci doivent être spécifiés dans la syntaxe suivante:

modeX=XON*$10000000+XOFF*$10000

Le manuel de référence Commodore vous informera plus amplement.

Autres commandes

SERIAL BUFFER *(Définit la taille du tampon série)*

SERIAL BUFFER Canal, Longueur

Affecte une "longueur" d'octets du tampon au canal demandé. Notez que la valeur implicite est de 512 octets et l'affectation minimum est de 64 octets. Il est préférable d'augmenter la taille du tampon pour les transferts à grande vitesse.

SERIAL FAST *(Active le mode de transfert RAPIDE)*

SERIAL FAST Canal

Cette instruction active un indicateur "rapide" spécial dans le périphérique courant et désactive beaucoup de contrôles internes qui ralentiraient le processus de communication. Utilisez-le pour les transferts à grande vitesse tels que MIDI. Attention: Lorsque vous appelez cette commande, le protocole sera modifié et indiquera:

PARITE PAIRE, NO XON/XOFF, 8 bits.

SERIAL SLOW *(Fait passer le transfert sériel sur mode LENT)*

SERIAL SLOW Canal

Rétablit le périphérique série sur la vitesse normale et réactive tous les contrôles d'erreur.

SERIAL CHECK *(Edite l'activité du périphérique série courant)*

=SERIAL CHECK (Canal)

Demande au périphérique de lister son état courant. Vous pouvez l'utiliser pour vérifier si toutes les informations que vous avez transférées à l'aide d'un appel précédent à la commande SERIAL SEND ont été envoyées.

CHECK=FALSE (0) ->	si la dernière commande série est toujours en train d'être exécutée.
CHECK=TRUE (1) ->	Bravo!

SERIAL ERROR *(Signale si le dernier transfert a abouti)*

=SERIAL ERROR (Canal)

Cherche l'octet ERREUR dans le périphérique série. Une valeur de zéro indique que tout est parfait. Le cas contraire indique que la dernière transmission n'a pas abouti.

Envoyer de longues chaînes

Transmettre une longue chaîne peut prendre beaucoup de temps, particulièrement si les vitesses de transmission sont lentes. Puisqu'AMOS exécute en alternance plusieurs

tâches, votre programme continuera à se dérouler APRES l'exécution de l'instruction SERIAL SEND.

Il est absolument important d'EVITER de provoquer une récupération de place avant la fin du transfert sinon vos données seront altérées.

Par conséquent:

Utilisez la fonction =SERIAL CHECK avant de faire beaucoup de travaux comportant des chaînes.

Effectuez une récupération de place en utilisant X=FREE pour vous assurer que votre programme n'en provoque pas un automatiquement.

Utilisez l'instruction SERIAL OUT canal,adresse,longueur en plaçant l'emplacement d'un banc de mémoire préalablement réservé dans "adresse".

Pour de plus amples informations

Vous trouverez de plus amples informations sur le système sériel d'Amiga dans le Manuel de référence ROM KERNEL de Commodore, Bibliothèque et Périphériques. Ceci permettra aux utilisateurs experts de profiter au maximum du périphérique série.

Code source

Vous trouverez le code source de l'extension et le fichier des équivalences d'AMOS Basic sur la disquette de mise à jour AMOS (PPD36).

Ce code a été assemblé avec Genam. Il vous faudra tous les fichiers ".I" de Commodore et le fichier AMIGA.LIB pour l'assembler.

Assemblez-le avec le type des caractères du symbole "dependant", le type de programme "linkable" et l'info débogage "export". Reliez-le avec: BLINK SERIAL.O To SERIAL.LIB AMIGA.LIB.

Si vous rencontrez des problèmes avec l'extension ou si vous découvrez des bogues, veuillez écrire à l'adresse ci-dessous en expliquant précisément ce qui s'est passé. Envoyez une copie de la portion du programme qui n'a pas abouti. Cela nous facilitera la vie!

AMOS Serial Enquiry,
Europa House,
Adlington Park,
Adlington,
Macclesfield,
SK10 4NP
Angleterre

24 Configuration d'AMOS

AMOS peut facilement être adapté à vos besoins, en changeant les messages d'erreur dans une autre langue, en configurant l'écran pour l'exploiter sous un système NTSC. Toutes les options qu'AMOS charge et exécute peuvent être modifiées au sein du programme de configuration (Config.AMOS).

Le programme de configuration AMOS

Une des fonctions les plus remarquables d'AMOS Basic est la possibilité d'adapter le logiciel précisément à vos besoins. L'utilitaire de Configuration AMOS vous donne un contrôle sans précédent sur les mécanismes du système AMOS. Il existe un très grand nombre d'options possible vous permettant de changer quoi que ce soit, du nombre maximal de bobs à l'écran à l'apparence précise des diverses fenêtres du Menu. Vous pouvez même renommer les messages d'erreur s'affichant au cours de votre programme! Toutes ces informations sont conservées dans un fichier spécial de configuration du dossier du système AMOS. Ces données sont automatiquement chargées à l'exécution d'AMOS.

En raison de la puissance même de cette utilitaire, il est absolument important de le traiter avec un peu de prudence. En pratique, en commettant une seule grave erreur, vous pourriez vous retrouver avec un système inutilisable! Nous vous recommandons vivement d'effectuer tous vos changements sur une copie de SAUVEGARDE de la disquette de programme AMOS. N'utilisez JAMAIS la disquette de programme AMOS d'origine à cette fin! Si vous avez tout mélangé, nous serions heureux de vous envoyer un nouveau fichier de la configuration mais vous serez privé de votre AMOS Basic pendant quelques jours.

Charger l'utilitaire de configuration

Charger l'utilitaire de configuration ne pourrait plus facile: Insérez une copie de SAUVEGARDE de votre disquette de programme AMOS dans le lecteur interne et chargez le fichier Config.Amos. Lorsque le programme est lancé, il vous sera proposé de développer le tampon de l'éditeur. Cliquez sur *OUI* ou appuyez sur la touche O pour continuer.

Vous pouvez alors faire fonctionner ce programme de la façon habituelle. Après que le programme ait été initialisé, un joli arc-en- ciel s'affichera à l'écran. Vous pouvez alors maintenir le bouton droit de la souris abaissé pour amener les options du menu.

Avant d'avancer plus loin, il vous faudra charger un fichier de configuration en mémoire. Les commandes LOAD/SAVE (CHARGER/SAUVEGARDER) se trouvent au menu du Disque. Si vous utilisez ce programme pour la première fois, sélectionnez l'option Charger la configuration par défaut pour charger les paramètres d'orgine se trouvant sur le disque.

Vous pouvez alors sauvegarder les changements effectués sur le disque en utilisant les options Sauvegarder Configuration ou Enregistrer Sous. Ces paramètres seront automatiquement installés au prochain chargement d'AMOS Basic. Voici une liste complète des diverses options du menu:

Menu AMOS

A propos de ce programme: Affiche un simple écran de titre.

Sortir: Sort d'AMOS Basic. Attention! Si vous n'avez pas sauvegardé votre configuration sur disquette, vous perdrez tous vos changements.
Se brancher au système: Quitte AMOS Basic et se branche sur le CLI ou le workbench.

Menu Disque

Charger configuration par défaut: Charge le fichier de configuration de la disquette programme AMOS. Celui-ci est conservé sous le nom d'AMOS1.2.Env dans le dossier AMOS_System.

Charger autre configuration: Charge un autre fichier de configuration à partir de la disquette. Il est parfaitement convenable d'avoir plusieurs fichiers de ce type sur votre disquette d'initialisation mais seul AMOS1.2.Env par défaut sera effectif.

Si vous êtes un fan du CLI, vous pourrez écrire un petit fichier séquentiel qui vous permettra de sélectionner une des diverses configurations lors de la mise en route. Il pourrait faire partie de votre séquence habituelle de la mise en route. A condition que vous gardiez toutes vos configurations dans le dossier AMOS_System, il vous suffit de donner un nouveau nom au fichier config. d'origine, tel que AMOS.EN1, et d'affecter le nom AMOS.ENV à votre fichier sélectionné.

Sauvegarder configuration: Sauvegarde les paramètres courants sur votre disquette programme. Ceux-ci ne seront pas fixés en mémoire lorsque vous chargez AMOS Basic. Ne changez JAMAIS la configuration de la disquette programme d'origine ou votre installation pourrait s'embrouiller à jamais!

Sauvegarder Comme: Conserve vos paramètres dans un fichier particulier. La configuration actuelle sera intacte.

Menu Configuration

Nouveau clavier: Ceci change le gestionnaire de clavier utilisé par AMOS Basic. Ces gestionnaires DOIVENT être créés à l'aide du définisseur spécial de clavier se trouvant sur la disquette programme AMOS. Les gestionnaires se trouvant sur la disquette des suppléments NE sont PAS autorisés. Vous trouverez une liste des gestionnaires disponibles dans le dossier KEYBOARDS de la disquette programme AMOS.

Extensions chargées: A la différence de ses nombreux concurrents, AMOS Basic est incroyablement facile à développer. Toutes les bibliothèques des nouvelles instructions peuvent s'ajouter directement au progiciel existant, en utilisant un système puissant de fichiers d'extension. Si vous êtes un programmateur de code machine, vous pourrez

même créer ceux-ci vous-même! Reportez-vous au code source se trouvant sur la disquette AMOS extras pour de plus amples informations.

Si vous ajoutez un nouveau fichier d'extension à votre système, vous devez informer AMOS de son emplacement exact sur la disquette. Pour cela, vous pouvez vous servir de la liste d'extensions AMOS. Il vous suffit de déplacer le curseur sur la zone appropriée et de cliquer sur le bouton gauche de la souris. Vous pouvez alors taper le nom intégral du chemin de votre extension et terminer l'opération avec un retour. Comme de coutume, vous êtes libre d'éditer toutes les chaînes en utilisant les touches habituelles du curseur.

Nom accessoire HELP: Cette fonction vous permet de changer le nom et la position de l'accessoire chargé lorsque vous appuyez sur la touche Help depuis l'éditeur. Les deux premières chaînes contiennent le nom de votre nouvel accessoire Help. Ces chaînes doivent toujours être les mêmes ou vous obtiendrez un message d'erreur lorsque vous essayez d'utiliser ce système.

Le dernier article de la liste contient le message qui sera affiché si vous ne trouvez pas l'accessoire sur la disquette courante.

Nom des fichiers système AMOS: Cette fonction vous présente une liste d'importants fichiers système utilisés par le package AMOS. Elles contiennent les données nécessaires aux icônes du programme AMOS, les curseurs de la souris, les paramètres de polices de caractères et les définitions de clavier. En changeant ces définitions, vous pouvez changer radicalement l'apparence de nombreux programmes Basic.

Lecteurs et ports: Cette fonction contient une liste complète de tous les périphériques pouvant être accédés depuis AMOS Basic. Nous y avons déja porté la plupart des périphériques. Mais si vous souhaitez y ajouter quelque chose d'inhabituel comme les interfaces midi, il vous faudra intégrer ces noms à la liste et ajouter les gestionnaires de périphériques appropriés au dossier devs.

Noms de touche fonction: Cette fonction vous permet de renommer toutes les options au menu utilisées par l'éditeur d'AMOS. Il vous suffit de déplacer la souris sur l'option et de cliquer sur le bouton gauche. Vous pouvez alors taper le nouveu nom de la commande sélectionnée. Si l'anglais est votre seconde langue, vous pourrez exploiter ce système pour créer une version spéciale de l'éditeur AMOS dans votre langue maternelle. Il existe un large éventail de possibilités pour laisser aller votre imagination si vous voulez un peu vous amuser.

L'effet de ces commandes de menu sera toujours le même, peu importe du nom que vous leur donnez!!!

Filtre négatif de répertoire: Cette fonction contient une liste de fichiers qui seront automatiquement omis de tous les listages de répertoires à venir. Ces fichiers peuvent comprendre des «jokers» si besoin; il est donc facile de supprimer toute une liste de fichiers à la fois. Reportez-vous à la commande AMOS SET DIR pour de plus amples informations.

Touches de fonction Amiga: AMOS Basic est livré avec un système puissant de

macros du clavier. Vous pouvez accéder à celles-ci en appuyant à la fois sur la touche Amiga de gauche ou de droite et sur une des autres touches de fonction. Voir la commande KEY$ pour de plus amples informations.

Par défaut, ces macros sont définies selon une série d'instructions Basic couramment utilisées. Vous pouvez les changer en ce que vous voulez ou presque!

Initialisation de l'éditeur: Ces options contrôlent la manière dont AMOS sauvegarde les fichiers sur la disquette, change la taille du tampon et la vitesse de défilement de la fenêtre de l'éditeur. Si vous avez l'intention d'utiliser AMOS du WorkBench, vous allez sans doute sauvegarder une icône avec vos programmes Basic. Une autre option utile est de décomprimer la taille du tampon du texte utilisé par l'éditeur. Ceci augmentera la longueur maximale des programmes Basic pouvant entrer dans votre ordinateur.

Initialisation de l'écran: Cette fonction définit la position et la taille des diverses zones d'écran, telles que la fenêtre de l'éditeur, la boîte du menu et l'écran en mode direct. En pratique, il est plus sage d'utiliser cette fonction avec prudence, puisque toutes les options ne sont pas toutes autorisées. Même si AMOS a ses limites et si vous essayez de modifier le format de l'écran de façon à le rendre méconnaissable, il y a peu de chances qu'AMOS marche.

Initialisation de l'interprèteur: Si vous disposez d'une mémoire suffisante, vous pouvez utiliser ces fonctions pour augmenter le nombre de bobs et changer la taille du tampon du sprite (voir SET SPRITE).

Palettes d'écran - Page 1: Cette fonction change toutes les couleurs utilisées par les fenêtres de l'Editeur AMOS. Avant de régler ce paramètres, il serait bon de sauvegarder les valeurs d'origine sur une disquette vierge en utilisant l'option Sauvegarder du menu du disque. Ceci vous permettra d'expérimenter autant que vous le souhaitez, sans courir le risque de créer un écran complètement inutilisable.

Vous savez sans doute que les boutons + augmentent le degré d'intensité du code couleur et boutons - l'abaissent. Les paramètres de la couleur courante sont indiqués par un tracé apparaissant lors du déplacement de la souris sur les options appropriées.

Palettes d'écran - Page 2: Cette fonction modifie les couleurs utilisées par la fenêtre en mode direct, la ligne d'erreur et l'écran Follow.

Définition d'AMOS selon le système NTSC: Cette option définit tous les écrans de façon à ce qu'ils s'inscrivent dans l'écran NTSC. C'est utile pour les utilisateurs américains puisqu'AMOS est configuré pour PAL.

Définition d'AMOS selon le système PAL: Cette option définit toutes les options d'écran selon la configuration PAL.

Menu messages

Ce menu vous permet de remplacer tous les messages produits par le système AMOS par vos *propres* versions améliorées. Toutes les options de ce menu fonctionnent de façon similaire: Pour éditer un message, déplacez le curseur de la souris sur la ligne

appropriée et cliquez sur le bouton gauche. Vous pouvez alors entrer votre chaîne de remplacement à partir du clavier et éditer le texte en utilisant les touches habituelles du curseur. Lorsque vous avez terminé, vous pouvez y placer le nouveau message en appuyant sur Retour.

A la gauche de la fenêtre du message se trouvent trois icônes. Les icônes fléchées vous permettent de parcourir les lignes de message. Vous pouvez revenir au principal écran de configuration au moyen de l'option Sortir.

De l'éditeur: Ces messages sont produits par l'Editeur AMOS et sont affichés sur la ligne d'information. Vers le bas de la liste se trouve une série de chaînes de caractères qui définissent l'apparence générale de la ligne d'information. N'hésitez pas à les éditer à votre gré.

De test: Si vous faites fonctionner un programme ou cliquez sur la commande TEST, AMOS effectue une vérification détaillée de la syntaxe Basic de votre programme. S'il rencontre des problèmes, un message d'erreur s'affiche. Vous pouvez changer ces messages de la même façon que l'option ci-dessus.

De l'exécution: En pratique, les erreurs ne sont pas toutes détectées par la vérification de la syntaxe initiale. Il existe donc une série particulière de messages d'exploitation qui est produite si AMOS détecte une erreur pendant que votre programme est en train de se dérouler.

Des operations disque: C'est un assortiment de messages qui est sorti lors du chargement et de la sauvegarde à partir de l'éditeur. Certains messages sont affichés sur la ligne d'information tandis que d'autres forment les titres des divers sélecteurs de fichier.

Il existe également une série de chaînes définissant les chemins de recherche des commandes charger, sauvegarder et fusionner. Par exemple, si vous importez des programmes en format Ascii depuis STOS ou Amiga Basic, il se peut que vous trouviez commode de transformer le nom du chemin *.ASC en *.*.

Du selecteur de fichiers: Vous voulez personnaliser le sélecteur de fichier AMOS? Eh bien, c'est l'option qu'il vous faut! Vous êtes libre de changer tous les boutons et les messages du programme de configuration. Vous pouvez même remplacer les flêches utilisées par les barres de défilement!

De l'écran d'erreur: Ce sont les messages affichés le long de la ligne d'erreur lorsqu'un problème se présente dans votre programme Basic.
De l'écran mode direct: Cette fonction spécifie l'apparence générale de la fenêtre de la touche de fonction affichée en mode direct.

Du titre de l'AMOS: Si AMOS est chargé en mémoire, un écran de titres s'affiche au centre de la fenêtre de l'éditeur. Cette option vous permet de personnaliser le titre à souhait.

Notez que AMOS annexe automatiquement une liste complète de toutes les extensions couramment installées à votre nouvel écran de titres.

Faire fonctionner AMOS depuis le CLI

Vous pouvez exécuter AMOS depuis le CLI si besoin. Il vous suffit de placer votre copie de la disquette programme AMOS dans votre lecteur et de taper:

```
cd amos:
amos 1.2
```

Il existe trois syntaxes possibles de cette commande:

AMOS 1.2
Exécute AMOS Basic et lance l'éditeur à la mise en route.

AMOS 1.2 nomfichier
Appelle AMOS, charge et exécute automatiquement le programme Basic sélectionné. La taille minimale du tampon de l'éditeur est automatiquement pré-définie pour conserver de la mémoire. Exemple:

amos 1.2 map_editor_amos (Lance l'éditeur de carte après le chargement d'AMOS)

AMOS 1.2 -memoire
Charge AMOS Basic et réserve le nombre nécessaire d'octets destinés à être utilisés par l'éditeur. Remarquez le signe moins avant le nombre. Par défaut, l'éditeur a une taille de 32.767 kilo-octets. Elle peut être augmentée en utilisant l'option Set Text du menu de recherche AMOS, d'une option au programme CONFIG. ou directement de la ligne de commandes. Exemple:

amos 1.2 -50000 (Lance AMOS et réserve 50k pour l'éditeur)

AMOS peut également être appelé à partir d'un fichier séquentiel ou pendant la Startup-Sequence dans le répertoire S: de votre disquette d'initialisation.

Faire fonctionner AMOS depuis le WorkBench

Initialisez le workbench de la façon habituelle et insérez la disquette programme AMOS dans un lecteur. Sélectionnez la disquette AMOS: en utilisant la souris pour afficher une liste des fichiers disponibles du programme. Vous pouvez alors lancer AMOS Basic directement en cliquant deux fois sur l'icône appropriée.

Attention! Cette procédure n'est pas recommandée sur un A500 non étendu, puisque le WorkBench consomme de très grandes quantités de mémoire et laisse presque rien pour vos programmes Basic. Il est bien plus logique de charger AMOS en initialisant votre Amiga, en plaçant la disquette de programme dans le lecteur zéro.

Charger un programme AMOS depuis le WorkBench

Vous pouvez allouer une icône à n'importe quel de vos programmes AMOS Basic. Vous pouvez activer cette fonction en utilisant une option spéciale de l'utilitaire de

configuration. Une icône sera créée pour chaque programme Basic que vous sauvegardez ultérieurement sur le disque. Lorsque vous sélectionnez cette icône du WorkBench, AMOS se chargera automatiquement en mémoire et votre programme Basic s'exécutera immédiatement.

Notez que les utilisateurs de disque dur devront réaliser quelques petits travaux de configuration avant d'utiliser ce système. Changez le Default Tool de l'icône programme contenu dans le dossier AMOS_Système.

Faire fonctionner AMOS depuis n'importe quel endroit du disque

AMOS fait quelques travaux de ménage avant de se charger complètement et de se présenter à vous. Une de ces tâches est de localiser tous les fichiers implicites, tels que la flèche de la souris, les polices de caractère par défaut, etc... La procédure et la logique qu'il utilise sont les suivantes:

1.Vérifiez si un écran PAL ou NTSC est utilisé dans le système courant.

2.En fonction du système, il cherchera les fichiers AMOS1_2_PAL.Env ou AMOS1_2_NTSC.Env.

3.Faute de trouver ce fichier, il cherchera le dossier AMOS_SYSTEM dans le répertoire résident du lecteur courant. S'il ne le trouve également pas, il n'essaiera plus de se charger et reviendra au WorkBench.

4.Si AMOS sélectionne Default.env au point 2, il examinera alors le fichier d'environnement pour savoir où se trouvent les autres fichiers. C'est en utilisant ce processus clé qu'il relocalise AMOS sur un disque.

Supposons que nous voulions installer AMOS dans un dossier appelé UTILS sur un disque dur. Nous devons d'abord copier les fichiers suivants de la disquette programme AMOS sur le chemin du disque dur Dh0.UTILS/

AMOS_System Un dossier contenant tous les fichiers implicites
AMOS1.2Le principal fichier AMOS

Les fichiers AMOS1_2_PAL.Env et AMOS1_2_NTSC.Env qui se trouvent dans le dossier AMOS_System doivent être retirés et installés dans le dossier UTILS.

AMOS_System
AMOS1.2
AMOS1_2_PAL.Env
AMOS1_2_NTSC.Env

Nous devons d'abord changer le fichier Default.Env afin qu'il informe AMOS de la location de tous les fichiers de données. Pour cela, vous devez initialiser une copie d'AMOS et exécuter le programme Config.AMOS. Il vous suffit de vous assurer que les

noms et chemins des fichiers implicites et des extensions sont correctes.

Dans notre exemple, tous les fichiers se trouvent à l'intérieur du répertoire AMOS_System. Assurez-vous que les extensions sont les suivantes:

1-AMOS_System/Music
2-AMOS_System/Compact
3-AMOS_System/Request
4-
5-
6-AMOS_System/Serial

Les noms de fichier par défaut doivent être ainsi définis:

1-AMOS_System/Def_Icon
2-AMOS_System/Mouse.Abk
3-AMOS_System/Mouse.Default.Font
4-AMOS_System/Mouse.Default.Key

Un point important à noter est que seul le nom du chemin supplémentaire est nécessaire. AMOS ajoute AMOS_System sur le chemin courant (dans ce cas Dh0:UTILS) pour trouver le chemin entier: Dh0:UTILS/AMOS_System/.

Les utilisateurs de disque dur trouveront ces renseignements très utiles et éviteront de fourrer le dossier AMOS_System dans le répertoire résident du disque dur. Toute personne mettant au point une application CDTV en AMOS aura également besoin d'utiliser cette fonction.

Les informations ci-dessus sont également valables pour le système d'exploitation RAMOS, à l'exception que RAMOS n'a pas besoin des fichiers PAL ou NTSC.

25 Le système d'exploitation

RAMOS est une version particulière run-only d'AMOS Basic vous permettant de faire fonctionner vos programmes indépendamment du progiciel AMOS.

A la différence d'AMOS Basic, RAMOS est un programme du Domaine Publique et peut donc être librement distribué avec n'importe quel de vos programmes Basic. Par conséquent, vous pouvez vendre légalement vos programmes à toute personne possédant un Amiga, même si elle n'a pas une copie de l'AMOS Basic.

En ce qui concerne vos programmes, RAMOS est presque similaire à AMOS Basic. Il existe simplement deux ou trois différences:

- Il n'y a pas d'éditeur! RAMOS effectue une commande automatique Fermer Editeur après être chargé. Si une erreur se produit ou Cntrl+C est enfoncé, RAMOS revient directement au WorkBench, sans afficher un message d'erreur.

- Il n'y a pas d'écran par défaut. Ceci vous permet d'initialiser vos écrans au commencement de vos programmes, sans avoir à afficher l'écran implicite en premier lieu. Si votre programme utilise en fait l'écran implicite, il vous faudra ajouter la ligne suivante au commencement de votre listage:

```
Screen Open 0,320,200,16,Lowres
```

Installer RAMOS sur un nouveau disque

Afin de distribuer vos programmes sous format run-only, il faut réaliser une disquette run-only en appliquant la procédure suivante:

- Formatter une disquette vierge en utilisant l'option Initialiser disquette du WorkBench

- Lancer AMOS Basic comme de coutume et utiliser la commande Set Text b. du menu de recherche pour que la taille du tampon du texte soit de 5000 octets. Ceci réservera la mémoire maximale nécessaire pour la procédure d'installation.

- Charger et fair fonctionner le programme d'installation RAMOS se trouvant sur la disquette AMOS des suppléments (RAMOS_Install.Amos). Il vous suffit alors de suivre les invites pour copier RAMOS sur votre nouvelle disquette.

- Finalement, sortez d'AMOS Basic et lancez le CLI. Utilisez la commande INSTALL pour rendre votre disquette amorçable. Par exemple:

```
install Df0:
```

Si vous êtes limité à un seul lecteur, il vous faut entrer la commande suivante:

install ?

A l'invite suivante, placez votre nouvelle disquette dans le lecteur et tapez:

DF0:

Créer une disquette d'initialisation RAMOS

Sauvegardez votre programme sur la disquette run-only sous le nom de AUTOEXEC.AMOS. Lorsque vous amorcez le système depuis ce disque, RAMOS se chargera automatiquement et lancera votre programme Basic.

Appeler un programme RAMOS depuis le WorkBench

Avant d'accéder à cette fonction, il faut demander à AMOS de créer une icône pour vos programmes lors de leur sauvegarde sur disque. Ce processus est très simple et doit seulement être effectué une seule fois:

- Chargez une COPY d'AMOS Basic et faites fonctionner l'utilitaire CONFIG.AMOS comme de coutume.
- Une fois que le programme s'est initialisé, entrez les paramètres par défaut de la disquette en utilisant l'option Charger Configuration Implicite du menu de la disquette.
- Appelez maintenant la fonction Installation Editeur du Set Menu et cliquez sur la boîte Sauvegarder Icônes.
- Sortez du menu principal et sélectionnez Sauvegarder Configuration. Le nouveau paramètre sera activé automatiquement lorsque vous chargez AMOS Basic.

Vous pouvez alors sauvegarder vos programmes directement sur votre disquette run-only. Lorsque vous lancez le WorkBench, chaque programme a sa propre icône dans la fenêtre du répertoire. En raison d'un (petit) bogue dans la version courante d'AMOS, toutes les icônes sont initialement placées dans le coin gauche de l'écran. Heureusement, il est assez facile de repositionner vos icônes à l'aide de la souris et de sauvegarder la nouvelle disposition en utilisant l'option Snapshot (du menu Spécial).

La dernière étape consiste à définir RAMOS en tant qu'outil implicite. Cliquez sur l'icône du programme approprié et sélectionnez l'option Info du menu WorkBench. Changez l'outil implicite en remplaçant :AMOS par :RAMOS et installez l'icône sur le disque à l'aide de l'option Sauvegarder.

Après avoir suivi cette procédure, vous pourrez charger vos programmes directement du WorkBench en cliquant deux fois sur l'icône demandée. Comme de coutume, vous pouvez passer de votre programme au WorkBench à tout moment en appuyant sur les touches Amiga+A.

Exécuter RAMOS à partir du CLI

Si vous êtes familier avec le CLI de l'Amiga, vous pouvez solliciter directement RAMOS depuis la ligne de commande. La syntaxe est la suivante:

ramos1.2 [program.amos]

program.amos étant le programme que vous souhaitez exécuter. Si vous l'omettez, RAMOS essaiera de charger un fichier appelé AUTOEXEC.AMOS à partir du répertoire courant.

L'exécution de RAMOS peut faire partie d'un fichier séquentiel si besoin. Un conseil: Définissez toujours le lecteur courant à votre disquette run-only à l'aide de la commande CD avant d'appeler RAMOS, sinon RAMOS ne pourra pas trouver les divers fichiers du système dont il a besoin pour se dérouler. Exemple:

```
Cd Df0:
Ramos1.2
Petit RAMOS.Env
```

Le fichier RAMOS1_2.Env peut être remplacé par une autre version au nom identique prenant moins de place. Ce fichier est conservé sur la disquette Extras dans un dossier appelé "Petit RAMOS". Si vous êtes presque à court de mémoire, utilisez ce fichier au lieu du plus grand fichier RAMOS.Env. Cependant, il existe quelques restrictions:

1 Petit RAMOS n'est pas accessible par Config.AMOS
2 A cause du point 1, seules les trois extensions pour lesquelles il est défini seront reconnues et il ne peut être lancé dans un dossier défini par l'utilisateur.

Notes RAMOS

Vous pourrez trouver les paramètres de configuration utilisés par RAMOS dans le fichier RAMOS.ENV du répertoire AMOS_System. Ce fichier ne comprend pas les messages d'erreur habituels; il est donc environ 7k plus petit que la version équivalente AMOS. Vous pouvez modifier ces paramètres en utilisant l'utilitaire CONFIG.AMOS se trouvant sur la disquette programme AMOS.

Attention! Par défaut, RAMOS essaiera automatiquement de fermer le WorkBench à son chargement. Il vous sera donc impossible de revenir sur le WorkBench ou sur le CLI après avoir mené à bien votre programme. Pour éviter l'apparition de ce problème, il vous faut changer l'option appropriée dans le fichier de configuration ARMOS.ENV. Vous pouvez la modifier en utilisant l'option Configuration Editeur du programme Configuration. Voir la documentation sur l'utilitaire de configuration AMOS pour de plus amples informations.

26 Les accessoires

L'Editeur de sprite AMOS

Comme vous le savez, AMOS Basic vous offre de nombreuses et puissantes fonctions pour manipuler les bobs et les sprites. Afin d'intégrer ces objets à vos jeux, vous avez évidemment besoin de méthodes de conception des images sprites à l'écran de votre ordinateur. C'est le but de l'Editeur de sprites AMOS. Vous pouvez également utiliser ce programme pour créer des icônes. Il vous suffit de tracer vos objets en tant que bobs et de les sauvegarder sur disque. Vous pouvez alors convertir vos objets directement en icônes en utilisant l'utilitaire Icon_Conv.Amos se trouvant sur la disquette de programme AMOS.

Notez que les fichiers d'image créés par l'éditeur peuvent aussi bien utilisés avec les sprites qu'avec les bobs. La seule importante différence entre ces objets est que les sprites sont limités à 16 couleurs. Pour plus de simplicité, toutes références ultérieures faites dans ce chapitre se rapporteront aux sprites et non aux sprites/bobs.

Charger l'Editeur de sprites

L'Editeur de sprites AMOS peut être exécuté en tant que programme normal ou en tant qu'accessoire.

Charger l'Editeur en tant que programme

Placez votre COPIE de la disquette programme AMOS dans le lecteur courant et chargez le fichier Sprite_Editor.Amos. Cliquez ensuite sur Run et après un petit moment, l'Editeur de sprites s'exécutera.

Utiliser l'Editeur en tant qu'accessoire

Afin de faire fonctionner l'Editeur AMOS en tant que programme accessoire, il vous faut le charger en utilisant l'option Charger Autres de la fenêtre de la commande (Shift/F7). Après l'avoir chargé en mémoire, vous pouvez le solliciter directement depuis l'Editeur en utilisant l'option Run Autres. Votre programme courant ne sera pas du tout affecté par cette procédure.

Incidemment, l'Editeur de sprites demande environ 10 secondes pour initialiser. Pendant cette période, il se peut qu'un écran vide se présente à vous. C'est tout à fait normal et vous n'avez pas à paniquer. Vous allez utiliser l'Editeur de sprites dans quelques secondes.

Utiliser l'Editeur de sprites

La plupart des fonctions de l'Editeur de sprites AMOS sont contrôlées par l'intermédiaire d'une série de simples icônes à l'écran. Vous pouvez accéder à ces fonctions en cliquant sur l'option désirée à l'aide du bouton gauche de la souris. Les seules grandes exceptions à cette règle sont les commandes utilisées pour charger et sauvegarder les banques d'images.

La touche L charge une banque de sprites AMOS en mémoire. Un sélecteur de

fichier AMOS standard apparaît alors à l'écran et il peut être utilisé pour sélectionner une banque de sprites de la façon habituelle.

Similairement, la touche S vous permet de sauvegarder vos nouvelles images de sprite sur le disque courant. Notez que cet éditeur est seulement capable de charger les banques de sprites selon le format standard AMOS .Abk.

Les commandes

L'écran est divisé en quatre zones principales:

La fenêtre des icônes

Elle est divisée en 5 rangées de 10 icônes. Chaque icône contrôle une seule fonction de l'Editeur de sprites. Pour exécuter une de ces options, cliquez sur l'icône appropriée à l'aide du bouton gauche de la souris.

La fenêtre d'édition

La principale zone de tracé se trouve dans la boîte d'édition à la gauche de l'écran. Tracer une image ne pourrait être plus simple. Il vous suffit de déplacer le curseur de la souris sur la boîte d'édition et de maintenir un des boutons de la souris abaissé. Chaque bouton de la souris peut être affecté d'une couleur particulière de la fenêtre palette. Si vous tracez en maintenant les deux boutons de la souris abaissés, la couleur de tracé va virer aléatoirement entre deux couleurs. Au fur et à mesure que vous créez votre objet, l'image se trouvant dans la fenêtre de visée s'actualisera automatiquement.

La fenêtre de visée

Cette fenêtre contient une copie de la taille intégrale du sprite que vous êtes en train de tracer. Les dimensions de cet objet peuvent être changées à tout moment en développant la zone de visualisation à l'aide du bouton gauche de la souris. Pour activer le nouveau paramètre,cliquez une seule fois sur le bouton droit.

La palette de couleur

La palette de couleurs contient une liste des couleurs possibles pouvant être utilisées pour votre tracé. Pour changer le couleurs, cliquez sur la teinte désirée à l'aide du bouton gauche ou droit de la souris. La nouvelle couleur sera alors utilisée chaque fois que vous abaissez le bouton approprié de la fenêtre d'édition.

Si la palette ne contient pas la teinte précise que vous voulez, vous pouvez changer les paramètres RVB en utilisant l'icône RVB en haut et en bas de la fenêtre de l'icône.

Les icônes de tracé (3ème et 4ème rangée du bas)

Tracé par points — Trace des points à la position actuelle de la souris.

Graphisme — Très similaire à Tracé, excepté que la fonction relie automatiquement chaque couple de points que vous êtes en train de tracer à l'écran en formant une ligne. L'apparence du tracé est généralement bien plus lisse.

Ligne — Trace une ligne directe entre deux points. L'appui du bouton gauche de la souris positionne le point de départ. En maintenant ce bouton abaissé et en déplaçant la souris, vous pouvez alors poursuivre le tracé de la ligne à la longueur désirée. Vous pouvez positionner cette ligne à l'endroit que vous souhaitez, en la collant à l'aide du bouton gauche. Vous pouvez abandonner cette procédure en appuyant sur le bouton droit de la souris.

Coloriage — Place des points au hasard autour du curseur. Cliquez sur le bouton gauche pour agrandir la zone coloriée à la taille désirée. La vitesse de coloriage peut être réglée en utilisant une option de la fonction Décoration.

Boîte — Comme avec Ligne, maintenez le bouton gauche abaissé pour que la boîte prenne forme. Relâchez ensuite ce bouton et déplacez votre objet à la position désirée. Vous pouvez alors fixer la boîte à l'aide d'un simple déclic sur le bouton gauche de la souris. Vous pouvez ensuite abandonner l'opération en appuyant sur le bouton droit de la souris. Vous pouvez tracer une autre boîte ou revenir au mode de tracé de ligne en cliquant sur les icônes Ligne ou Tracé par Points.Boîte remplieldentique à la commande Boîte normale, excepté que la boîte est remplie avec le motif de remplissage courant qui peut être changé à tout moment en utilisant les icônes - et +.

Ellipse — Ellipse vide. Les dimensions de cet objet sont fixées en déplaçant le curseur de la souris tout en abaissant le bouton gauche. Après avoir défini votre ellipse, vous pouvez la coller à l'écran en cliquant une nouvelle fois le bouton gauche. Comme de coutume, le bouton droit vous permet d'abandonner l'opération et de forcer les dimensions de l'ellipse à zéro.

Texte Vous permet d'ajouter un petit texte à votre sprite en utilisant la police de caractère couramment définie. Vous pouvez taper ce texte à partir de votre clavier. Il peut être édité à l'aide de la touche d'espacement arrière et de le coller à l'endroit désiré en cliquant sur le bouton gauche de la souris.

Remplissage Remplit les formes irrégulières. Si vous appuyez sur le bouton gauche, le remplissage continue jusqu'à ce qu'il rencontre une couleur différente de celle sélectionnée pour le remplissage (coloration).

Les fonctions Barre et Remplissage utilisent les motifs de remplissage couramment définis. Vous pouvez les changer en cliquant sur les icônes - et +. Notez également qu'avec les fonctions Boîte, Barre, Ellipse, Ligne, Texte et Coloriage, vous pouvez changer la taille du pinceau en cliquant sur le bouton droit.

Couper/Coller

En haut et à gauche de l'écran se trouvent deux icônes vous permettant de couper toute section de la fenêtre d'édition et de la coller à un autre endroit sur le sprite courant. L'opération Couper est représentée par une paire de ciseaux.

Après avoir sélectionné cette commande, vous devez d'abord positionner la souris dans la zone du coin supérieur gauche. Si vous relâchez ce bouton, cette zone se chargera en mémoire pour être utilisée en tant que pinceau. Pour copier cette zone, il vous suffit de cliquer sur le bouton gauche de la souris à la nouvelle position. Le bouton droit efface la zone se trouvant sous le bloc et le motif de remplissage courant qu'elle contient.

Après avoir terminé votre collage, vous pouvez cliquer sur une des icônes de dessin pour continuer votre dessin. Le bloc actuel sera conservé en mémoire et peut être réactivé en sélectionnant l'icône Coller.

A la droite des icônes Couper et Coller se trouve l'icône Transparence. Il vous permet d'afficher la couleur zéro à l'écran lors de vos collages.

La série suivante d'icônes vous permet de changer la position de la "poignée" du pinceau couramment sélectionné. Elles peuvent être utilisées avec les commandes Boîte, Barre, Ligne, Ellipse et Coller.

Point chaud

Sous les icônes Couper/Coller se trouve une série de huit icônes qui contrôlent la position du point chaud de votre sprite. Le point chaud est utilisé comme point de référence lors du calcul de la position de votre sprite à l'écran. Par conséquent, s'il se trouve au centre du sprite et si le sprite est placé aux coordonnées 23,40, le centre du sprite se situera aux coordonnées 23,40.

Les sept premières icônes déplacent le point chaud vers l'une des positions prédéfinies. Les diverses options de gauche à droite sont: Haut Gauche, Haut Milieu, Haut Droit, Bas Gauche, Bas Milieu, Bas Droit, Centre.

La dernière icône vous permet de positionner le point chaud à n'importe quel endroit du sprite. Si le curseur se trouve sur les icônes point chaud, le point chaud courant s'affichera dans la fenêtre d'édition.

Dimensionner un sprite

Vers le coin supérieur droit de la fenêtre Icône se trouvent quatre icônes vous permettant de changer le format du sprite que vous éditez. La première portant l'inscription Voir Sprite vous montre l'image courante, exactement comme elle serait affichée par la commande AMOS.SPRITE. (Les sprites sont plus rapides mais ils sont limités à 16 couleurs et par d'autres restrictions à l'écran. Voir le manuel AMOS pour plus d'informations).

Les trois autres icônes sont:

Redimensionner — Amène le clavier de curseurs à l'écran pour vous permettre de changer la taille du sprite. Le facteur Zoom de la fenêtre Edition change automatiquement au fur et à mesure que la taille du sprite est changée. Appuyez une nouvelle fois sur le bouton droit pour sortir de ce mode. La taille du sprite peut être également modifiée en déplaçant la zone d'affichage à l'aide du bouton gauche de la souris et en appuyant sur le bouton droit lorsque le sprite a atteint la taille désirée.

Optimiser — Réduit la taille du sprite au minimum exigé par l'image (économise de la mémoire).

Réduction — L'image est poussée en haut et à gauche du sprite, aussi loin que possible.

Note: — Si la taille d'un sprite est de 32 par 32 pixels, il peut être agrandi quatre fois et être affiché dans son intégralité. En dessus de cette taille, une fenêtre de 32 par 32 pixels s'affichera multipliée par 4. La zone affichée peut être changée à l'aide des touches du curseur. La zone que vous éditez est indiquée par deux bornes sur la zone d'affichage du sprite.

Autres fonctions d'édition

A la droite des icônes de tracé se trouvent plusieurs fonctions utiles servant à modifier l'image:

CLR — Efface l'image en utilisant les couleurs/motifs courants.

Bascule — H/V Bascule l'image horizontalement ou verticalement.

Rotation — Fait tourner l'image sur 90 degrés dans le sens des aiguilles d'une montre.

DéfilementFait défiler l'image dans la fenêtre d'édition (le bouton droit vous permet de sortir).

Zoom Vous permet de réduire ou agrandir l'image. Dans ce mode, vous pouvez redimensionner l'image en cliquant sur la boîte d'affichage à l'aide du bouton gauche de la souris. Si vous appuyez sur le bouton droit, vous figez l'image et sa taille actuelle. Comme de coutume, sélectionnez tout autre mode de tracé pour abandonner cette fonction.

REZ Vous permet de changer la résolution de l'écran et le nombre de couleurs avec lesquelles vous travaillez. L'éditeur supporte actuellement les modes d'écran suivant:

Faible Res: modes 8, 16, 32 et 64(EHB)
Haute Res: modes 2, 4, 8, 16

Saisir les sprites

L'objet que vous éditez peut être transféré vers et depuis la banque de sprites en utilisant une série d'icônes se trouvant en bas de l'écran. A la gauche de ces icônes se trouve le bouton RVB qui vous permet de changer les valeurs RVB de toutes les couleurs de la fenêtre de la palette. Vous verrez à son côté les icônes de saisie qui sont dans l'ordre suivant:

Insérez sprite Crée un espace et met le sprite dans la banque.

Mettre sprite Met l'image courante directement dans la banque de sprites, écrasant le sprite qui s'y trouvait.

Saisir sprite Saisit un sprite de la banque et vous permet de l'éditer.

Effacer sprite Efface un sprite de la banque de sprites (Pas de droit à l'erreur!)

Supprimer banque Efface TOUS les sprites de banque (Pas de droit à l'erreur!).

Il existe cinq icônes en bas et à droite de la fenêtre icône vous permettant de parcourir tous les sprites disponibles en mémoire:

|< Le curseur se déplace sur le premier sprite en mémoire.

< Le curseur se déplace vers la définition précédente du sprite.

Nombre Entre le mode visualisation et affiche trois sprites à la fois. Pour sortir de ce mode, il vous suffit de déplacer le curseur en bas de la zone d'icônes.

>	Le curseur se déplace sur le sprite suivant de la liste courante.
>|	Le curseur se déplace sur le dernier sprite en mémoire.

Lorsque le curseur se trouve sur un de ces icônes, le sprite sélectionné s'affichera dans la fenêtre de visualisation. Ce sprite n'affectera pas l'objet que vous êtes en train d'éditer.

Autres icônes

Annuler	Recrée la dernière petite portion de tracé/zoom/rotation/bascule, etc...
Sortir	Sort du programme
Décoration	Vous permet de changer les couleurs de contour de la fenêtre d'édition, de modifier le son "ping" et le volume lorsque vous cliquez sur une icône. Elle vous permet également de changer la vitesse de coloriage.

Raccourcis clavier

Il existe une autre solution remplaçant l'utilisation des icônes habituelles: vous pouvez accéder à une gamme d'opérations depuis le clavier. En appuyant sur les touches du curseur en maintenant la touche Shift abaissée, vous pouvez effectuer les déplacements suivants:

Shift haut	Mémorise l'image à l'emplacement courant de la banque (identique au bouton Mettre).
Shift bas	Saisit le sprite courant de la banque (identique au bouton Saisir).
Shift gauche	Mémorise l'image à l'emplacement courant puis saisit le sprite précédent de la banque.
Shift droit	Mémorise l'image à l'emplacement courant puis saisit le sprite suivant de la banque.

En outre, les touches de fonction vous donnent un accès rapide aux 10 sprites suivants dans la banque de la mémoire courante. Par exemple, si le pointeur du sprite indique le sprite 5, F1 solliciterait le sprite 5, F2 le sprite 6 et ainsi de suite jusqu'à F10 qui sélectionnerait le sprite 14.

Pour mémoriser un sprite dans la banque, appuyez sur Shift et une touche de fonction. Pour le retrouver, appuyez simplement sur la touche de fonction.

Remarques sur la mémoire

Le programme fonctionne parfaitement sur un A500 non étendu. Si vous avez désespérément besoin d'une mémoire plus étendue pour créer de plus grands nombres de sprites, vous pouvez contourner le problème en supprimant les routines Décoration (Et non pas les routines du bouton). Vous pouvez également sauvegarder un peu de mémoire en modifiant la routine de la banque des images (banque 6).

Des informations plus détaillées sur la façon d'exploiter au mieux une machine d'1/2 meg paraîtront dans le bulletin du club AMOS. Le premier numéro comprend un article intitulé A500 Blues. Si vous n'êtes donc pas déjà un membre du club AMOS, cela vaut la peine d'y adhérer immédiatement!

Le découpeur de sprites

Le découpeur de sprites vous permet de saisir une liste d'images de sprite à partir d'une image de format IFF. Cela signifie que vous pouvez créer vos objets en utilisant votre progiciel préféré de dessin et les sauvegarder directement dans une banque de mémoire AMOS normal.

Au commencement du programme, le sélecteur du fichier AMOS se présentera à vous. Il est utilisé pour choisir une image et la charger à partir de la disquette courante.

L'utilisation de ce découpeur de sprites est exceptionnellement simple. Toutes les options sont contrôlées par une petite barre de menu contenant sept icônes. Cette barre peut être déplacée en haut de l'image à l'aide des touches du curseur. Vous pouvez également utiliser ces touches pour faire défiler une portion de votre image dans la fenêtre d'affichage.

Pour sélectionner une option, il vous suffit de cliquer sur l'icône appropriée à l'aide du bouton gauche de la souris. Voici une explication complète des diverses icônes du menu:

-/+ Choisit le numéro de l'image que vous souhaitez saisir. Si l'image a déjà été définie,une copie s'affichera le long de la barre du menu. Attention! En raison d'un petit bogue dans la version courante du Découpeur de sprites, l'image peut ne pas s'afficher par erreur. Pour remédier à ce problème, revenez sur mode direct et relancez immédiatement le programme du Découpeur de sprites. Vos nouvelles images ne seront pas du tout affectées par cette manoeuvre.

Icône ciseaux Saisit une portion rectangulaire de l'écran et la charge dans l'image courante. Lorsque vous sélectionnez cette option, la barre du menu disparaît immédiatement vous permettant d'accéder à l'intégralité de l'écran.

Vous pouvez alors saisir votre image de la mémoire en utilisant une boîte d'agrandissement normale. Positionnez le curseur sur le coin supérieur gauche de votre image et maintenez le bouton gauche de la souris abaissé. Au fur et à

mesure que vous déplacez la souris sur l'écran, un cadre entourera l'image actuelle. Lorsque vous êtes satisfait de votre nouvel objet, relâchez le bouton de la souris pour charger cette image en mémoire.

Disque à écran Charge une image IFF à partir du disque.

Disque à .ABK Charge une banque de sprite AMOS existant à partir du disque. Attention! Cette option détruira toutes vos images existantes. Notez également qu'une nouvelle palette de couleurs sera chargée directement depuis la banque de sprites.

ABK à disque Sauvegarde vos images sur le disque sous la forme d'une banque de sprites normal. Cette banque peut être ultérieurement chargée directement dans un de vos programmes Basic AMOS.

SortirSort du Découpeur de sprites et revient sur AMOS Basic.

Total AMOS Map Editor (TAME)

Lorsque vous écrivez un jeu d'arcade, il est souvent nécessaire de produire des douzaines d'arrière-plans d'écran différents pour votre programme. La pratique a montré que la façon la plus efficace de créer ces écrans est de les construire à partir d'un certain nombre de petites tuiles ou briques. Ceci vous permet de conserver chaque écran en tant que simple liste de ses composants et ceci vous économise ainsi une quantité considérable de précieuse mémoire.

Afin de créer un de ces écrans, vous avez évidemment besoin d'un certain éditeur de cartes. C'est le but du **T**otal **A**MOS **M**ap **E**ditor (TAME). C'est un utilitaire de dessin puissant vous permettant de concevoir sans effort les écrans dont vous avez besoin dans vos jeux. Un exemple parfait de ces cartes en action est l'excellent jeu Magic Forest que vous trouverez sur la disquette de données AMOS.

Une fois que vous avez créé vos cartes, vous pouvez les afficher à l'écran en utilisant les procédures TAME spéciales se trouvant à la fin de la démo Magic Forest. Vous pourrez également les afficher directement à l'aide du système d'extension TOME qui va bientôt sortir.

Lors de la mise en route, l'Editeur de cartes vous demande de choisir une image IFF en utilisant un sélecteur de fichier standard. L'image se charge en mémoire, divisée en une série de titres qui seront utilisés pour construire vos cartes. Les titres sont alloués de gauche à droite, depuis l'angle supérieur gauche de l'image.

Par défaut, la carte sera remplie automatiquement d'une copie de la tuile numéro zéro. Si vous voulez démarrer avec un écran vide, vous devez vous assurer que l'angle supérieur de votre image source est entièrement vierge.

L'écran TAME est divisé en trois principales zones.

La ligne du Menu

La ligne du menu est indiqué par un grand A en haut de l'écran. En sélectionnant cette ligne à l'aide du bouton droit de la souris, vous pouvez appeler une gamme d'options puissantes au menu. Celles-ci vous offre une multitude de fonctions utiles vous permettant de définir vos cartes.

Lorsque la souris est positionnée sur la ligne du menu, une fenêtre d'information s'affichera à l'écran.

La fenêtre d'information

Elle contient de nombreux renseignements utiles concernant votre carte:

Tuile Contient une copie de la tuile qui a été selectionnée en tant que pinceau. Notez que les couleurs de la tuile peuvent être très différentes de celles utilisées sur l'écran réel.

ECRAN Affiche le nom et le type d'écran courant.

CARTEL' Editeur de cartes AMOS est parfaitement capable de gérer des cartes qui sont bien plus grandes que la zone visible de l'écran. Toutes les opérations de tracé ont lieu dans une fenêtre sur la carte courante.

A la droite de la ligne d'information se trouve un grand rectangle contenant une barre verte. C'est en fait l'article d'un menu pouvant être manipulé pour contrôler la position précise de la fenêtre se rapportant à la totalité de la carte. Il vous suffit de maintenir le bouton gauche abaissé et de déplacer la barre à l'aide de la souris. La fenêtre se déplacera en conséquence. Vous pouvez également faire défiler la carte en utilisant un joystick branché dans la prise de gauche ou en appuyant sur les touches fléchées du clavier du curseur.

Tp LftX: Représente l'abscisse de la fenêtre courante se rapportant au coin supérieur gauche de la mappe. Ces abscisses sont mesurées en unités de tuile.

Y: Indique l'ordonnée Y de la portion de la carte que vous êtes en train d'éditer.

Taille Affiche la taille totale de votre carte. X: Largeur de la carte en tuiles. Y: Hauteur de la carte en tuiles.

La fenêtre de tracé

La fenêtre de tracé occupe la plus grande partie de la zone visible de l'écran. Par défaut, la tuile courante est automatiquement affectée à la souris. Elle peut être collée à

tout endroit de l'écran à l'aide d'un simple déclic sur le bouton gauche de la souris.

La sélection d'une tuile est facile: Déplacez la souris sur la fenêtre de tracé et appuyez sur le bouton droit de la souris. La carte sera immédiatement remplacée par l'écran de titres. Vous pouvez alors positionner la souris sur la tuile désirée et revenir sur la fenêtre de tracé en utilisant le bouton gauche de la souris.

Les options du menu

Le menu du fichier

Charger Tuiles — Vous permet de charger un nouvel écran IFF que vous pourrez utiliser comme tuiles. Vous pouvez également saisir une image qui a été comprimée à l'aide de la commande AMOS SPACK.

Charger Carte — Charge une nouvelle carte à partir du fichier .MAP du disque.

Sauvegarder Carte — Sauvegarde la carte courante en tant que fichier .MAP

Sortir — Sort de l'Editeur de cartes et revient sur AMOS Basic.

Menu des blocs

Couper bloc — Saisit une portion de votre écran et la sauvegarde dans une zone temporaire du menu. Cliquez d'abord sur la flèche **T L** se trouvant dans l'angle supérieur gauche de la zone à découper et placez ensuite la flèche **B R** sur le point diagonalement opposé. Vous pouvez alors découper cette zone en cliquant simplement le bouton gauche de la souris.

Après avoir créé un bloc, le mode coller sera activé automatiquement. Vous pouvez donc copier un bloc n'importe où sur la carte en cliquant de nouveau le bouton gauche de la souris.

Coller bloc — Colle un bloc préalablement créé à l'écran. Déplacez le curseur sur la fenêtre de tracé et cliquez sur le bouton gauche de la souris pour coller le bloc à la position souhaitée.

Menu de Modification

Taille des tuiles — Vous permet de changer la taille des tuiles. Les tailles possibles sont les suivantes: 16x16, 16x32, 32x16 et 32x32.

Taille de la carte Règle la taille de la carte courante en unités de tuile.

Menu d'Edition

Tracé Maintenez le bouton gauche de la souris abaissé pour tracer une tuile à la position courante du curseur.

Boîte Comme avec Couper, entrez les dimensions de votre boîte en cliquant sur les coins supérieurs et inférieurs de la zone qui sera remplie de la tuile courante.

Choisir Tuile Vous permet de choisir une tuile du jeu courant. Vous pouvez sélectionner cette option sur la fenêtre de carte en appuyant sur le bouton droit.

L'Editeur de menu

Une des fonctions les plus remarquables d'AMOS Basic est son système de menu étonnamment puissant. Il comprend de nombreuses fonctions qui seraient impressionnantes sur un poste de travail graphique le plus perfectionné.

Malheureusement, lorsque vous définissez vos menus, il est facile de se retrouver submergé par le nombre incroyable de commandes possibles. Vous allez être vraiment tenté d'éviter les complications et de simplifier vos menus autant que possible.

L'Editeur de menu AMOS contourne habilement ce problème en vous permettant de concevoir directement vos menus en utilisant la souris. Ces menus peuvent être sauvegardés sur disque sous la forme d'une banque de mémoire AMOS. Vous pouvez alors charger cette banque dans vos programmes Basic et accéder à l'intégralité du menu à l'aide d'une simple instruction Basic.

A l'aide de l'Editeur de menu AMOS, il est facile d'expérimenter. Il n'y a donc rien qui vous arrête d'intégrer des menus hallucinants dans vos propres programmes. L'Editeur de menu est contrôlé par l'intermédiaire d'une série de simples icônes d'écran et toutes les options peuvent être exécutées en sélectionnant le bouton approprié et en cliquant une fois sur le bouton gauche de la souris.

Charger l'accessoire du menu

L'Editeur de menu peut être chargé en tant qu'accessoire ou en tant que programme Basic normal. S'il est installé en tant qu'accessoire, il saisira automatiquement tous les bobs, icônes ou menus utilisés par votre programme courant.

Le menu du système

A la mise en route, le principal menu du système se présentera immédiatement à vous. Il supporte une plage d'utiles commandes du système pour charger, sauvegarder et initialiser vos menus. Comme nous l'avons mentionné précédemment, toutes les options sont en vos mains. Il vous suffit de déplacer la flèche sur l'icône appropriée et de la sélectionner en appuyant sur le bouton gauche de la souris.

Voici les fonctions des commandes du système:

Créer un nouveau menu

Définit un nouvel écran servant d'arrière-plan de menu. Le format de cet écran peut être saisi à l'aide de la souris. Vous pouvez alors avancer sur la principale fenêtre d'édition pour créer divers articles au menu.

Editer le menu courant

Saute directement à l'écran Editeur et initialise le menu couramment sélectionné.

Sauvegarder la banque

Créer une banque de mémoire pour vos du menumenus et la sauvegarde sur disque. Ce menu peut être chargé dans votre programme en utilisant la ligne suivante par exemple:

```
Load "nomfichier.abk"
```

Vous pouvez initialiser votre nouveau menu en tapant:

```
Bank to menu 6
```

Vous pouvez alors activer ce menu en utilisant MENU ON.

Notez que ces banques de menu sont permanentes et seront automatiquement sauvegardées avec vos programmes Basic. Par conséquent, une fois que vous avez installé votre menu et écrit le gestionnaire de menu en AMOS Basic, vous pouvez en fait l'oublier.

Charger un menu

Charge une définition de menu à partir d'un fichier .MENU du disque. Ce fichier doit avoir été préalablement défini en utilisant l'option courante Sauvegarder. Après avoir chargé le menu, le programme passera au principal écran de l'Editeur.

Sauvegarder le menu courant

Sauvegarde l'intégralité de la définition de votre menu dans un fichier séquentiel .MENU. Ce fichier peut seulement être chargé depuis l'Editeur de menu. Si vous voulez accéder à votre menu depuis AMOS Basic, vous devez utiliser l'option Sauvegarder Banque du Menu.

Sortir et saisirRevient sur AMOS

Basic. Si vous lancez l'Editeur en tant qu'accessoire, le nouvelle banque de menu sera automatiquement installé. Le menu que vous avez

créé peut être directement sollicité depuis votre programme courant.

Sortir — Sort et revient sur AMOS Basic. Attention! Cette commande efface toute la définition de votre menu!

La fenêtre de l'Editeur

La fenêtre de l'Editeur est divisée en deux sections. En haut de l'écran se trouve le menu que vous êtes en train d'éditer. Chaque article du menu est représenté par une petite boîte numérotée. Ces boîtes sont disposées de la façon suivante:

- Tous les articles du niveau d'un menu sont disposés dans une même colonne verticale. Par conséquent, la première colonne indiquera la ligne de titre de votre menu et la seconde détaillera les divers articles du menu.
- Chaque article est strictement numéroté en fonction de sa position dans la hiérarchie du menu. Par exemple, l'article 1.2 indiquera la deuxième option du menu du titre numéro 1.

Bien que la disposition ci-dessus puisse sembler un peu compliquée au premier abord, vous pourrez lire toute la structure d'un menu d'un seul coup d'oeil avec de la pratique. **Un conseil:** Cliquez sur le bouton droit de la souris pour tester le menu que vous êtes en train d'éditer. Cela vous permettra de comparer le numéro affiché à la ligne du menu réel qu'il représente.

Si votre menu est vraiment grand, il peut dépasser les dimensions de la fenêtre de l'Editeur. Dans ce cas, vous pouvez parcourir toute la définition du menu en utilisant les icônes fléchées se trouvant à la gauche de l'écran.

Le menu de l'Editeur

Le menu de l'Editeur contient une liste de commandes vous permettant de définir la structure physique de votre menu. Toutes les options sont exécutées sur l'article du menu courant. Celui-ci peut être sélectionné directement depuis la fenêtre de l'Editeur en utilisant la souris et il sera mis en vidéo inverse. Le menu est divisé en trois principales zones:

Menu d'état de l'article

Ces options changent l'état de l'article sélectionné au menu. Voir le Chapitre 16 du manuel AMOS qui vous donnera une liste détaillée des diverses fonctions du menu. Notez que les options Tline/Line/Bar et Br.Move/Br.Sta affectent toute une branche de votre menu et pas seulement l'article courant!

Active/Inact — Spécifie si l'article courant doit être actif. Seuls les objets actifs peuvent être sélectionnés depuis le menu terminé.

Line/Bar/T.line — Organise le niveau courant du menu selon Line, Bar ou Total Line.

Linked/Separ. Vous permet de décider si chaque article de votre menu est relié à un autre se trouvant au-dessus de lui. Voir la commande AMOS MENU LINKED qui vous donnera toutes les explications de ce système.

Br.sta/Br.mov Abbréviation de Branch stationary (Branche fixe) et Branch moveable (Branche mobile). L'action de cette commande est identique à un appel lancé aux instructions MENU MOVABLE (MENU MOBILE) ou MENU STATIC (MENU STATIQUE) d'AMOS Basic. Il spécifie si la branche courante du menu peut être repositionnée par l'utilisateur.

It.sta/It.mov Abbréviation de Item stationary/Item moveable (Article fixe/Article mobile). Cette fonction vous donne la possibilité de redisposer individuellement tous les articles d'un menu. Voir les instructions Basic MENU ITEM MOVEABLE/STATIC pour de plus amples informations.

Hiérarchie du menu de l'Editeur

Ces icônes vous permettent de développer la hiérarchie de ce menu et d'ajouter de nouveaux articles à votre menu courant. Toutes les options d'édition normales sont prévues, y compris Ajouter, Insérer et Effacer.

Après avoir créé votre menu, vous pouvez le tester en appuyant sur le bouton droit de la souris de la façon habituelle.

Si les articles sont mobiles, vous pouvez aisément les repositionner à l'écran en utilisant la souris. Les nouvelles positions seront conservées à moins que vous effaciez un article de la branche actuelle ou utiliser la commande Redéfinir les Positions de Menu. Dans ce cas, le menu sera entièrement réinitialisé.

Ajouter article Ajoute un nouvel article à la fin du menu courant. Par défaut, l'article sera automatiquement affecté à la chaîne du texte EMPTY. Ceci peut être modifié en utilisant une des nombreuses options du menu Tracé.
Ins itemInsère un nouvel article du menu à la position actuelle du curseur.

Branche Crée une branche du menu au point existant et ajoute un nouvel article à cette branche.

Effacer Efface un article du menu.

Redéfinir la position

Du menuRéinitialise les coordonnées de tous les articles appartenant à la branche actuelle du menu.

Le menu du tracé

Le menu du tracé offre une porte d'accès au puissant système de tracé à l'écran qui peut peut être utilisé pour dessiner les articles de votre menu. Il existe quatre options possibles:

Normal — Définit l'apparence normale d'un article de votre menu.

Mise en valeur — Définit la forme de votre article après qu'il ait été sélectionné à l'aide de la souris.

Inactive — Trace l'objet qui sera affiché si l'article est désactivé par la commande AMOS MENU INACTIVE.

Arrière-plan — Spécifie l'arrière-plan de votre article. Cette zone sera toujours tracée derrière l'article courant du menu.

L'écran de tracé de l'article

Il est au coeur de l'Editeur de menu puisqu'il vous permet de définir l'apparence précise de chaque article du menu affiché à l'écran.

Tous les articles du menu sont composés d'une liste de 21 éléments graphiques. Ceux-ci correspondent aux instructions emboîtées que l'on trouve dans une chaîne de menu AMOS normale. Chaque élément peut être édité indépendamment ce qui facilite beaucoup votre travail de modification des menus que vous avez créés.

Supposez que vous vouliez créer un article du menu comprenant un texte simple. Dans ce cas, le premier élément de la liste serait alloué à une boîte vide et le second affecté au texte de votre menu.

Le menu des éléments

-/+ — Ces icônes vous permettent de choisir l'élément que vous allez définir à l'écran. Vous pouvez voir une image de cet élément dans la fenêtre courante des éléments.
InsInsère un nouvel élément graphique à la position actuelle. Le 20ème élément sera poussé à l'extérieur et sera donc définitivement perdu!

Del — Efface l'élément courant.

Pousser — Sauvegarde un élément dans une zone de mémoire temporaire.

Coller — Copie une image de la mémoire dans l'élément courant. Ces données doivent avoir été préalablement conservées en utilisant la commande POUSSER.

L'écran de travail

Toutes les opérations de tracé ont lieu sur un écran de travail indépendant. Celui-ci contient une image de l'intégralité de l'article comme elle apparaîtrait au menu final. Vous pouvez examiner votre nouveau menu à tout moment en déplaçant simplement la souris sur les icônes de tracé et en appuyant sur le bouton droit de la souris.

Tracer un élément

Vous devez d'abord choisir la couleur de votre élément à partir de la fenêtre de la palette. L'icône 1 représente la couleur d'encre et l'icône 2 spécifie le paramètre de papier courant.

Vous pouvez changer l'encre en déplaçant la souris sur la moitié supérieure de la palette et en cliquant sur le bouton gauche de la souris lorsque vous avez choisi votre nouvelle couleur. Similairement, la couleur papier peut être modifiée en sélectionnant une couleur parmi celles de la moitié inférieure de la fenêtre de la palette. Notez que la couleur de contour prend automatiquement la couleur de l'encre courante.

Après avoir fait votre choix de couleurs, vous pouvez alors tracer votre élément en utilisant une des puissantes fonctions du menu de tracé.

Le processus de tracé est très facile. Déplacez le curseur de la souris sur la fenêtre de l'article courant et maintenez le bouton gauche abaissé. L'intégralité de l'objet sélectionné peut alors être déplacée à l'aide de la souris. Lorsque vous relâchez le bouton de la souris, l'objet que vous avez créé sera automatiquement affecté au pinceau courant. Il peut être alors collé à la position souhaitée en cliquant de nouveau sur le bouton gauche de la souris.

Si vous commettez une erreur, vous pouvez effacer le paramètre courant du pinceau en appuyant sur le bouton droit de la souris alors que la flèche se trouve sur la zone de travail.

Il est important de comprendre que chaque élément de l'article de votre menu peut seulement être affecté à un seul objet du menu de tracé. N'oubliez donc pas d'augmenter le numéro de l'élément avant de l'ajouter à votre conception ou votre élément existant sera entièrement remplacé.
Voici une liste des options de tracé proposées:

Line (Ligne)Trace une ligne dans la couleur d'encre courante.

Box (Boîte)Affecte l'élément graphique courant à une boîte vide.

Bar (Barre)Trace une barre pleine dans le motif de remplissage courant. Le motif peut être changé en utilisant les boutons + du menu Paramètres.

Ellipse Crée une ellipse ou un cercle vide.

Icône Colle une icône sur l'article du menu. Le numéro d'image peut être sélectionné en utilisant les touches fléchées de gauche et de droite. Pour utiliser cette fonction, la banque d'icône doit avoir été préalablement chargée en utilisant l'option Charger Banque de Mémoire du menu du disque.

Bob Identique à l'icône excepté que l'image est prise directement de la banque de sprites.

Texte Saisit le texte à partir du clavier et l'affecte à l'élément courant du menu.

T.Len Fixe la longueur maximale du texte de votre menu.

Make Inverse Inverse les couleurs du texte et papier dans l'élément actuel.

Make Border Entoure l'objet d'un joli cadre.

Le menu des paramètres

Ce menu vous permet de modifier l'état des options texte et graphiques. Les trois icônes "n" vous permettent de choisir les modes texte comme Gras, Souligné et Italique. Dès que vous cliquez sur l'une de ces icônes, le texte de la fenêtre des polices de caractères se modifiera aussitôt.

-/+ Sélectionne un motif de remplissage pour la commande Barre. Le motif courant est affiché à la droite de la fenêtre de polices de caractères. Si vous avez préalablement chargé la banque de sprites, cliquez sur l'icône - pour affecter une image sprite au motif courant de remplissage.

Contour Ajoute un contour à vos motifs de remplissage.

S.Font Charge la liste des polices de caractères depuis la disquette programme et vous permet de choisir un nouveau style de caractère.

Le menu Misc

Pousser article (Push Itm)Sauvegarde l'intégralité d'un article dans une zone de mémoire temporaire. Ceci vous permet multiplier les copies de vos objets favoris.

Coller article (Paste Itm)Remplace l'article courant et le contenu de la zone mémoire créée en utilisant PUSH ITM.

Charger une banque de mémoire Charge une série d'icônes ou de bobs à partir du disque. Après avoir installé la banque, la nouvelle palette sera automatiquement copiée dans l'arrière- plan de l'écran.

L'éditeur AMAL

AMAL est un nouveau langage d'animation étonnant qui vous permet de produire sans effort des déplacements lisses comme dans un jeu d'arcade moderne, sans avoir à recourir au complexe code machine.

La vitesse globale d'exécution de ce système est phénoménale! AMOS peut facilement exécuter 16 séquences individuelles d'animation AMAL simultanément sans perturber vos programmes Basic normaux. En pratique, il existe deux méthodes possibles d'intégration des routines AMAL dans vos jeux. L'approche habituelle est de placer toutes les commandes AMAL dans une variable en chaîne. Vous pouvez alors exécuter vos séquences d'animation en utilisant une instruction AMAL comme suit:

AMAL n,A$

Bien que ceci convienne aux petites routines, vous vous apercevrez qu'il est vraiment difficile de l'appliquer aux longues séquences d'animation. Nous avons donc conçu un accessoire de l'éditeur AMAL vous permettant de comprimer tous les programmes que vous utilisez fréquemment dans une seule banque de mémoire. Vous pouvez alors lancer vos routines AMAL à partir du clavier et les éditer à l'écran en utilisant les touches du curseur habituelles.

Le débogage de vos programmes AMAL ne pourrait être plus simple puisque l'éditeur est livré avec son propre système intégré dans le moniteur. Ceci vous permet d'exécuter un programme, instruction après instruction. Tous les registres AMAL sont directement accessibles depuis l'écran; il est donc facile de localiser précisément une erreur.

Une autre fonction très intéressante est la possibilité de créer des schémas compliqués de déplacement à l'aide de la souris. Ces derniers peuvent être affectés à n'importe quel objet de votre choix et rejoués en utilisant une seule instruction Play AMAL.

Après avoir créé une banque, vous pouvez solliciter vos nouvelles séquences d'animation à partir d'AMOS Basic en appelant une commande spéciale AMAL. La syntaxe de cette instruction est la suivante:

AMAL n,p

où *n* est le numéro du canal AMAL que vous souhaitez affecter et *p* représente le numéro d'un programme AMAL conservé dans la banque courante de la mémoire AMAL.

En réalité, il faut voir la puissance étonnante du système AMAL en fonction pour y croire. Il n'y a rien qui vous arrête d'écrire 90% de vos routines de jeu directement en AMAL et de vous servir d'AMOS pour réaliser les routines ennuyeuses d'initialisation et les tableaux de marquage des points.

Si vous sortez quelque chose de vraiment spécial, n'oubliez pas d'envoyer une copie à Mandarin Software. Nous attendons beaucoup de choses de ce système et aimerions voir un jeu entièrement écrit en AMAL.

Charger l'éditeur AMAL

L'éditeur AMAL peut être utilisé en tant qu'accessoire ou en tant que programme Basic normal. S'il est installé en tant qu'accessoire, il saisira automatiquement une copie des banques de mémoire définies par votre programme Basic.

L'éditeur de chaîne

L'éditeur de chaîne vous propose une multitude de fonctions vous permettant de créer des chaînes de programme AMAL et de les conserver dans la banque de mémoire AMAL. L'écran est divisé en trois principales zones.

La ligne d'informations

Affiche le mode courant et le numéro du programme AMAL que vous êtes en train d'éditer. C'est également l'emplacement de la barre de menu utilisée pour contrôler l'Editeur. Ce menu est accessible en utilisant le bouton droit de la souris de la façon habituelle.

La fenêtre de sélection

Comme son nom le suggère, cette fenêtre vous permet de sélectionner un des divers programmes AMAL conservés en mémoire. Afin de comprendre le fonctionnement de cette fenêtre, vous devez lire les numéros affichés de haut en bas. Le premier article de la liste est EE (Environment Editor - étudié ultérieurement), le deuxième article est le canal 00, le troisième canal 01, etc... Le programme actif est toujours mis en valeur en couleur inversée.

Pour éditer un de vos programmes

AMAL, il vous suffit de cliquer sur le numéro du canal approprié à l'aide du bouton gauche de la souris. La fenêtre de l'Editeur se chargera immédiatement, accompagnée de la définition courante si elle existe. Vous pouvez également sélectionner le programme depuis le clavier en maintenant à la fois une des touches de l'Amiga enfoncée et la flèche gauche ou droite du curseur.

Notez que si l'option Synchro Off a été activée à partir du menu des options, le nombre des canaux disponibles passera de 16 à 63. Toutefois, seules les 16 premières routines peuvent être exécutées en utilisant le système d'interruption AMAL normal.

La fenêtre de l'Editeur

La fenêtre de l'Editeur fonctionne de la même façon que l'éditeur du programme AMOS; vous devez donc vous sentir immédiatement à l'aise avec son utilisation. Vous pouvez lancer chaque programme AMAL directement depuis le clavier et l'éditer en

utilisant les touches normales du curseur. Tous les caractères sont entrés à la position courante du curseur. Vous pouvez déplacer ce curseur dans votre programme en utilisant la souris.

Il existe également une série complète de commandes Couper/Coller dans le menu Bloc. Elles facilitent l'affectation d'une même séquence d'animation à diffèents objets à l'écran.

Voici une liste des commandes possibles:

Retour	Insère une ligne
Cntrl-Y	Efface une ligne
Tab	Saute directement au prochain taquet de tabulation.
Flèches	Déplace le curseur d'une position dans la direction appropriée.
Shift-Flèches	Déplace le curseur au début/fin de la ligne ou en haut/bas de l'écran.

Déplacer l'écran d'édition

Si l'écran de l'éditeur perturbe l'affichage de votre programme existant, vous pouvez aisément le repositionner à l'aide de la souris. Déplacez le curseur de la souris sur une des barres noires de déplacement encadrant l'écran et maintenez le bouton gauche enfoncé. L'écran peut être déplacé vers le haut ou vers le bas si besoin. Notez que l'écran de l'Editeur est toujours affecté du numéro d'écran sept. N'utilisez donc pas cet écran dans vos programmes Basic.

Exécuter vos programmes AMAL

Après avoir lancé vos programmes AMAL en mémoire, vous pouvez les exécuter en utilisant une des gammes d'options du menu d'édition. Voici une petite liste des fonctions les plus importantes:

EXECUTER tout	Appelle la chaîne d'environnement AMAL (étudiée ultérieurement) et exécute vos programmes AMAL simultanément.
EXECUTER courant	Initialise l'écran et exécute le seul programme AMAL que vous êtes en train d'éditer. Après avoir démarré votre programme, vous pouvez revenir sur la fenêtre de l'Editeur à tout moment en frappant une touche.
Déboguer	Lance le débogueur AMAL.

Les commandes de l'environnement

En pratique, il est rare que les programmes AMAL soient exécutés en étant complètement isolés. La plupart des routines AMAL demande une prudente initialisation depuis AMOS Basic avant de pouvoir être utilisées.

L'éditeur AMAL comprend donc une version abrégée de l'interpréteur AMOS Basic connu sous le nom de Générateur d'Environnement. Celui-ci vous permet de réaliser toutes vos routines d'initialisation directement depuis l'éditeur AMAL sans avoir à revenir constamment sur AMOS Basic.

Bien que ces commandes soient similaires à leurs équivalents en AMOS Basic, nous avons été obligés de simplifier certaines instructions afin de conserver de la mémoire. Voici une liste complète des commandes supportées par ce système:

SPRITE OFF	Efface tous les sprites à l'écran
BOB OFF	Efface tous les bobs
RAINBOW	DELEfface tous les arc-en-ciel
'	REMark au début d'une ligne
SCREEN OPEN	Comme AMOS
SCREEN DISPLAY	Comme AMOS
SCREEN OFFSET	Comme AMOS
SCREEN	Comme AMOS
SCREEN CLOSE	Ferme tous les écrans
SCREEN CLONE	Comme AMOS
DOUBLE BUFFER	Comme AMOS
DUAL PLAYFIELD	Comme AMOS
DUAL PRIORITY	Comme AMOS
LOAD IFF "nom",screen	Vous DEVEZ donner un numéro d'écran!
COLOUR	Comme AMOS
GET SPRITE PALETTE maskVous	DEVEZ indiquer le type de masque
FLASH	Comme AMOS
FLASH OFF	Comme AMOS
SET RAINBOW	Comme AMOS
RAINBOW	Comme AMOS
LOAD "nom",bank Vous	DEVEZ spécifier le nom de la banque
ERASE banque	Comme AMOS
BOB	Comme AMOS
SET BOB	Comme AMOS
SPRITE	Comme AMOS
SET SPRITE BUFFER	Comme AMOS
HIDE ON	Comme AMOS
UPDATE EVERY	Comme AMOS
CHANNEL TO SPRITE	channel,sprite
	Différent à partir de là!
CHANNEL TO BOB	channel,bob
CHANNEL TO SCREEN DISPLAY ch,scr	
CHANNEL TO SCREEN OFFSET ch,scr	
CHANNEL TO SCREEN SIZE ch,scr	

CHANNEL TO RAINBOW ch,r
SET REG number,value Définit le registre AMAL A-Z (0- 25)

Il existe également un certain nombres d'instructions Test bien utiles. La syntaxe générale de ces commandes est la suivante:

INSTRUCTION condition : statements

Les instructions suivant la condition seront seulement exécutés si la condition est VRAIE. Si la condition est FAUSSE, ces commandes seront rejetées et le programme sautera immédiatement à la ligne suivant de votre routine d'Environnement.

IF SCREEN number	(Vrai si l'écran a été OUVERT)
IF NOT SCREEN number	(Vrai si l'écran est à presént FERME)
IF BANK number	(Vrai si la banque a été réservée)
IF NOT BANK number	(Vrai si la banque n'est pas réserve)
IF REG number,value	(Vrai si Reg A-Z (0-25) égale une valeur)

Vous pouvez lancer le programme d'environnement de l'éditeur AMAL, comme s'il était un programme AMAL normal. Il vous suffit de cliquer sur l'option EE dans la fenêtre de sélection et de taper vos procédures depuis le clavier. Votre nouveau programme sera automatiquement sauvegardé dans une partie de la banque courante de la mémoire. Il s'exécutera chaque fois que vous exécutez un programme AMAL de l'éditeur.

Attention! Si vous souhaitez appeler vos routines AMAL à partir d'un programme Basic, il vous faudra convertir vos commandes d'environnement en AMOS Basic puis ajouter les lignes appropriées au début de votre code. Ce processus devrait être peu important.

Note: Si vous utilisez des bobs, vous DEVEZ prévoir une instruction SCREEN pour diriger les données de sortie vers votre écran ou les bobs seront entraînés vers l'écran de l'éditeur (numéro 7).

L'Editeur "play"

L'Editeur "play" vous permet de saisir un schéma de déplacement compliqué depuis l'écran. Il échantillonne la position du curseur de la souris de façon répétée au fur et à mesure que ce dernier traverse l'écran et place ces données dans la banque de mémoire AMAL. Vous pouvez utiliser ce schéma ultérieurement pour animer pratiquement tous les objets que vous souhaitez.

Vous pouvez accéder à cette fonction en sélectionnant l'option Play du menu Edition. Comme l'Editeur de chaîne, vous pouvez modifier le positionnement de l'écran à votre gré.

Enregistrer un échantillon

Enregistrer un schéma de déplacement est facile. Choisissez d'abord le numéro d'échantillon que vous souhaitez définir en mettant en valeur l'article approprié de la fenêtre de sélection.

Appelez maintenant l'option Enregistrer du menu de déplacement. Vous serez

immédiatement invité à indiquer la vitesse courante d'enregistrement. Celle-ci établit une attente en 50ème de seconde entre chaque échantillon successif de votre nouveau schéma de déplacement.

Il existe une relation réciproque et inhérente entre la vitesse d'enregistrement et la quantité de mémoire consommée dans la banque AMAL. Les effets d'animation les plus lisses demandent un taux élevé d'échantillonnage et il est préférable de les créer en utilisant les paramètres par défaut. D'autre part, si vous avez l'intention de produire simultanément un grand nombre de schémas, vous pouvez vous en tirer avec un paramètre légèrement plus bas. Cela pourrait vous économiser des quantités considérables de mémoire précieuse.

Après avoir entré la vitesse, vous êtes prêt pour votre enregistrement. Déplacez la souris au point de départ de votre séquence d'animation et appuyez sur Retour. Les déplacements de votre souris seront alors constamment échantillonnés jusqu'à ce que vous sortiez de ce mode en appuyant sur le bouton gauche de la souris.

Ne soyez pas déçu si vous ne produisez pas immédiatement des résultats parfaits. Il y a un truc bien précis à saisir pour réaliser cette procédure et il se peut que vous ayez besoin d'expérimenter un peu pour obtenir les meilleurs résultats.

Tester vos schémas de déplacement

Lorsque vous avez défini une séquence de déplacement, vous pouvez la rejouer en utilisant la fonction Lecture de l'Editeur Play.

Vous pouvez également tester vos schémas aux côtés des objets réels que vous animez dans votre jeu. Pour cela, revenez sur l'éditeur de chaîne et entrez l'instruction suivante:

PL n

où
n est le numéro de la séquence de déplacement à jouer.

Rappelez-vous d'initialiser vos objets en utilisant le programme d'environnement avant d'exécuter cette commande ou l'instruction PL sera totalement ignorée.

Récapitulatif des commandes Jeu

Menu d'édition

Retour à l'éditeur de chaîne Revient sur le principal éditeur

Menu de déplacement

EnregistrementEnregistre un schéma de déplacement et le conserve dans la banque AMAL.

Lecture Rejoue une séquence pré- enregistrée en utilisant le curseur de la souris.

Insérer — Insère un nouveau schéma à la position courante. Tous les schémas se trouvant après ce point sont déplacés d'une position vers la droite. Attention! Le 48ème schéma de cette liste sera perdu!

Effacer — Efface le chemin courant de déplacement. Tous les schémas supplémentaire seront déplacés d'une position vers la gauche.

Le Moniteur AMAL

Le Moniteur AMAL comprend une gamme de fonctions puissantes qui simplifient vraiment le déboguage. Vous pouvez y accéder en sélectionnant Déboguer au principal menu d'édition.

A l'entrée, il vous sera présenté un écran semblant compliqué. Vers le haut de l'écran se trouve une fenêtre de sélection contenant une liste complète des programmes couramment actifs. Par défaut, tous les programmes AMAL en mémoire s'exécuteront automatiquement. Si vous voulez exécuter quelques-uns de ces programmes, vous devrez désactiver les routines indésirées manuellement depuis la fenêtre de sélection. Rappelez-vous: les programmes actifs sont mis en couleur inversée.

Changer un registre

Des registres externes: A la gauche de la fenêtre du moniteur se trouve une liste des 26 registres externes. Ces derniers peuvent être modifiés à tout moment en cliquant simplement sur le registre approprié et en entrant une nouvelle valeur en hexadécimal.

Les registres internes: Chaque programme AMAL a son propre jeu indépendant de registres internes. Ceux-ci sont affichés dans la petite boîte au centre de la fenêtre du moniteur. Le format de ces registres est le suivant:

R(numéro de canal,registre,numéro)

Vous pouvez étudier les registres internes utilisés par chaque programme AMAL en cliquant sur une des icônes fléchées se trouvant en bas du registre affiché. Pour définir un registre, sélectionnez-le avec le bouton gauche de la souris et entrez simplement une nouvelle valeur à partir du clavier (utiliser le format hexadécimal).

Déboguer vos programmes

Les principales options de déboguage sont contrôlées par un jeu de cinq boutons se trouvant à la droite de l'écran. Toutes les commandes peuvent être exécutées en sélectionnant une icône à l'aide de la souris ou en frappant la touche appropriée.

Avant d'utiliser une de ces options, vous devez d'abord initialiser le système AMAL en cliquant sur l'option Init ou en appuyant sur la touche I du clavier. Ceci fait fonctionner la routine d'environnement et prépare le moniteur. Vous pouvez alors tester votre programme en utilisant les commandes suivantes:

R (Run) — Exécute les programmes AMAL sélectionnés jusqu'à l'appui d'une touche.

G (Go until) Exécute vos programmes AMAL et revient au moniteur lorsque le registre choisi contient la valeur spécifique. Des points de rupture peuvent être appliqués en affectant une valeur spécifique au registre aux points appropriés de votre programme AMAL. Par exemple:

```
Let RZ=-1
```

Note: Si vous utilisez une notation hexadécimale, n'oubliez pas de précéder vos valeurs de test du symbole $.

S (Single Step) Exécute un seul pas de votre programme AMAL. Fonctionne également avec la commande Play AMAL.

(Esc) Revient à la principale fenêtre de l'éditeur.

Notes: Si la fenêtre du moniteur gêne l'affichage de votre programme, elle peut être repositionnée à l'aide de la souris. Déplacez-la simplement en utilisant les barres de déplacement se trouvant en haut et en bas de l'écran. Tous les sprites machine seront affichés devant la fenêtre du moniteur.

L'accessoire du lecteur Ascii

Les instructions du lecteur Ascii

AMOS évolue constamment et de nouvelles fonctions sont constamment ajoutées. Elles sont documentées dans une série de fichiers ASCII sur la disquette programme. Pour vous faciliter la vie, nous avons doté ce progiciel d'un puissant lecteur ascii. Il vous permet de lire (et d'imprimer) tous les fichiers de documentation directement depuis AMOS Basic. Vous pouvez charger le lecteur AMOS ascii depuis la disquette programme AMOS en appuyant sur la touche SOS. Avec un peu de chance, vous êtes maintenant en train de lire ce fichier en utilisant le programme du lecteur. Vous avez donc déjà compris les fonctions de certaines des icônes d'écran. Voici une explication complète des diverses commandes:

Le logo AMOS

Sort de l'accessoire du lecteur ascii et vous renvoie à l'écran de l'Editeur AMOS.

L'icône TV/moniteur

Le lecteur ascii affiche par défaut tous les fichiers de documentation dans le format d'écran européen PAL. Si vous êtes un utilisateur américain et possédez un écran NTSC, vous pouvez cliquer sur cette icône pour réduire la taille de la fenêtre du texte pour que celle-ci entre entièrement dans l'écran. La lettre "N" apparaîtra à l'intérieur de l'icône du moniteur pour mettre en valeur le nouveau paramètre.

L'icône du disque

C'est l'icône la plus importante. Elle charge un fichier de texte à partir du disque et l'affiche à l'écran de l'Amiga.

L'icône du fichier d'impression

Si vous avez une imprimante, vous pouvez utiliser cette icône pour obtenir une copie papier de n'importe quelle partie du document courant. Attention! Pour imprimer un document depuis ce programme, vous aurez besoin d'au moins 1 mégaoctet de mémoire. Ce n'est pas dû à AMOS mais à la taille énorme du lecteur de périphérique "IMPRIMANTE" de l'Amiga.

Cependant, si vous essayez d'imprimer à partir d'une machine dotée d'une mémoire de 512k seulement, une boîte d'alerte apparaîtra vous informant que vous disposez d'une mémoire insuffisante pour commencer à imprimer.

Les icônes fléchées ascendante/descendante

Ces deux icônes s'affichent seulement après avoir chargé un fichier du texte. Elles vous permettent de vous déplacer verticalement dans le document.

flèche ASCENDANTE: Affiche la page précédante
flèche DESCENDANTE: Affiche la page suivante du document

L'icône de liaison d'écran

L'icône d'écran apparaît dans le coin supérieur droit de l'écan. Elle est utilisée pour créer le menu SOS qui s'affiche automatiquement au chargement un fichier comportant l'extension ".lnk". (Examinez les fichiers de liaison disponibles sur la disquette programme.) Si vous cliquez sur une des options du menu, le lecteur ascii chargera immédiatement le fichier du texte qui a été affecté à cette option. Vous pouvez alors parcourir le document comme si vous l'avez lancé vous-même. Vous pouvez rappeler le menu à tout moment en appuyant simplement sur l'icône d'écran.

Note: Lorsque vous chargez un menu Liaison, vous devez toujours affecter le répertoire à la RACINE de votre disque courant en utilisant l'option SETDIR du sélecteur de fichier. Si vous oubliez, vous obtiendrez un message avertisseur d'erreur de disque lorsque vous essayez de sélectionner une option du menu.

Si vous voulez créer vos propres menus HELP, vous pouvez utiliser le puissant créateur de fichier de liaison prévu sur la disquette de programme AMOS. Il est donc facile de créer des menus HELP pour toutes vos applications AMOS Basic.

Lire de longs fichiers

Les utilisateurs AMOS disposant d'une mémoire d'un ou plusieurs mégaoctets peuvent augmenter la taille des fichiers que le lecteur ascii peut traiter, en utilisant la commande SET BUFFER. Le tampon de la variable doit être augmenté selon la formule suivante:

(Taille de fichier maximale*2)+1024k.

L'accessoire créateur de fichiers de liaison

Le Créateur de Fichiers de Liaison AMOS (Link File Creator) vous permet de créer de petits menus avec le lecteur Ascii. Pour cela, il vous suffit de taper une série de titres de menu et de les "relier" à un fichier de texte.

Au lieu d'énumérer toutes les options du menu, nous allons vous montrer comment créer un simple menu.

1ère étape: Sélectionnez l'option "CREATE LINK FILE" depuis le menu. Le programme vous demandera alors le titre de votre nouveau menu. Tapez "C'EST UN MENU TEST" et appuyez sur retour.

2ème étape: Vous verrez alors une jolie diposition de clavier à l'écran. Le curseur est positionné sur la boîte "OPTION TITLE". Tapez "C'EST L'OPTION 1" et appuyez sur retour. Un sélecteur de fichier va alors apparaître à l'écran vous demandant de sélectionner le nom d'un fichier de liaison. Sélectionnez le fichier que vous souhaitez charger lorsque l'option est sélectionnée à partir du lecteur ascii. Pour le moment, choisissez n'importe quel fichier que vous voulez.

3ème étape: Bien, vous devez maintenant avoir un joli menu comportant un titre, une seule option définie et sept autres options non définies. Chaque option peut être éditée en cliquant dessus et vous serez renvoyé à l'écran OPTION TITLE que nous vous avons montrez précédemment. Vous pouvez répétez cette procédure pour définir toutes les options de votre menu. Simple huh? Vous pouvez également changer le MENU TITLE de la même façon. Sélectionnez-le à l'aide de la souris et tapez votre nouveau titre à l'aide du clavier.

4ème étape: Après avoir créé votre menu, il vous faut le sauvegarder sur disquette.

Appelez les options SAVE ou SAVE AS du menu EDIT. Un sélecteur de fichier vous sera alors présenté. Tapez le nom du fichier comme de coutume. N'oubliez pas de suppléer le nom du fichier de l'extension .LNk.

Conseils et suggestions

Le tout premier créateur de fichiers de liaison a été conçu principalement pour donner une structure d'enchaînement à une série de courts fichiers de texte. Les longs fichiers de texte ralentissent le lecteur ascii et sont susceptibles d'envoyer le lecteur potentiel dormir!! Deuxièmement, il est préférable de ne pas toucher au Créateur de Fichiers de Liaison avant d'avoir taper tous vos fichiers de texte dans un traitement de texte. Vous saurez alors comment nommer les en- têtes de menu, etc...

Si vous avez le goût de l'aventure, vous pouvez enrichir l'éditeur de liaison en ajoutant par exemple de jolis graphismes aux menus ou en concevant une option DELETE OPTION. Si vous réalisez des changements utiles, veuillez les envoyer à Mandarin Software. Après tout, ils peuvent être suffisamment bons pour les intégrer à une nouvelle version AMOS.

Le définisseur de clavier

AMOS a prévu un définisseur de clavier perfectionné vous permettant de changer la disposition de tout votre clavier au moyen de quelques déclics sur le bouton de la souris. Il vous sera demandé de taper un nom de fichier pour sauvegarder ou charger une disposition de clavier: ce sera la seule fois quc toucherez le clavier pendant le cours d'utilisation de ce programme.

Lorsque vous déroulez le programme, une image du clavier de votre Amiga s'affichera. Les utilisateurs de certains pays remarqueront deux ou trois touches supplémentaire à l'écran. Ceci s'explique simplement: Certains claviers Amiga ont davantage de touches!

Toutes ces touches peuvent être redéfinies pour convenir à vos besoins, à l'exception de la touche RETOUR, la touche ECHAPPEMENT et la touche RAPPEL ARRIERE.

Etudions un exemple: Créons une nouvelle disposition de clavier!

1ère étape: Cliquez sur l'image de la touche Alt à l'écran à l'aide du bouton gauche de la souris. Le clavier sera alors retracé et la touche Alt sera mise en valeur. Il se peut que vous vous demandiez pourquoi un petit carré est dessiné sur certaines des touches. C'est pour vous indiquer que la touche n'a pas de vraie valeur Ascii (la touche n'a pas de caractère affichable).

2ème étape: Cliquez sur la boîte qui contiendrait normalement la touche 1 (en haut et à gauche du clavier). Les informations suivantes seront affichées dans un mince rectangle vers le haut de l'écran:

CODE DE POSITION DE TOUCHE (KEY SCANCODE): Contient le code de position de la touche mise en valeur.

VALEUR PAR DEFAUT (DEFAULT VALUE): Elle provient de la disposition du clavier qui est automatiquement saisi lors du premier déroulement du programme.

VALEUR COURANTE (CURRENT VALUE): Indique la valeur Ascii qui sera renvoyée à l'appui de cette touche du clavier.

3ème étape: Il nous faut sélectionner le caractère que nous souhaitons affecter à la touche ALT-1. Pour cela, cliquons sur les boutons FLECHES se trouvant juste au-dessus du clavier. Déplaçons la souris sur le bouton fléché de DROITE et maintenons le bouton gauche de la souris abaissé jusqu'à ce qu'un A majuscule apparaisse dans la boîte CARACTERE. (Valeur Ascii 65).

Vous pouvez affecter cette valeur en cliquant sur la boîte CARACTERE (vous entendrez un son avertisseur).

4ème étape: Enfin, vous pouvez tester votre affectation en déplaçant la souris sur la ligne d'informations de l'écran. C'est la ligne contenant toutes les informations sur la définition de la touche. (Elle commence par CODE DE DEFINITION DE TOUCHE:(KEY SCANCODE))

Cliquez sur cette icône à l'aide du bouton gauche de la souris et un petit curseur noir apparaîtra dans la boîte. Vous pouvez alors tester votre nouvelle touche en appuyant sur la touche "Alt" + la touche "1". Un "A" majuscule devrait apparaître.

Les options du Menu

Vous pouvez amener une ligne de menu à tout moment en maintenant le bouton droit de la souris abaissé. Le menu EDITER contient deux ou trois simples commandes qui vous permettent de charger et sauvegarder les différentes dispositions de clavier sur disquette. Il supporte également les options RESTAURER TOUCHES (RESTORE KEYS) et RESTAURER CLAVIER (RESTORE KEYBOARD) qui permettent de réinitialiser une ou plusieurs touches. Il peut être très utile pusiqu'il vous permet de corriger vos erreurs sans effort.

Si vous ne résidez pas en Grande-Bretagne et si vous avez crée une nouvelle disposition convenant à votre clavier, veuillez l'envoyer à Mandarin Software. Avec un peu de chance, nous pourrons sortir une disque du domaine publique comportant les nouvelles dispositions des claviers de tous les pays du monde (au moins pour ceux qui possèdent une Commodore Amiga!)

Générateur de banque d'échantillons

Ce programme vous permet de charger des échantillons en mémoire afin que vous puissiez constituer une banque de mémoire à utiliser dans vos programmes AMOS.

Mais pourquoi devons-nous choisir cette méthode? Qu'avons-nous à reprocher au format standard IFF? Eh bien, le système est très bien mais il n'est en fait pas très efficace en termes de mémoire. Puisque les échantillons sont souvent énormes, il est logique de réduire les besoins en mémoire au strict minimum. Cela vous permettra de condenser de plus nombreux et plus grands échantillons dans vos programmes AMOS.

Comment l'utiliser?

Le générateur de banque d'échantillons est entièrement contrôlé par le menu. Comme d'habitude, le menu apparaît seulement lorsque vous maintenez le bouton droit de la souris abaissé. Il est préférable de sélectionner les sons que vous souhaitez utiliser dans votre banque avant de lancer ce programme, puisqu'il n'est pas possible de jouer les échantillons directement depuis l'utilitaire. Ceci économise de la mémoire et permet aux utilisateurs disposant de 512k d'utiliser environ 40k pour leurs échantillons. NOTE: Les utilisateurs ayant une mémoire plus étendue devraient agrandir la taille de leur tampon au moyen de la commande DEFINIR TAMPON (SET BUFFER) au début du programme sinon ils seront limités aux 40k pré-définis.

Pour constituer une banque, il vous suffit de charger chaque échantillon en mémoire en utilisant l'option "CHARGER ECHANTILLON" (LOAD SAMPLE) de l'option EDITER (EDIT). Le programme reconnaîtra les données brutes de l'échantillon Amiga (ou un "DUMP" comme le programme Perfect Sounds l'appelle) et les échantillons créés avec le Maestro STOS. Si l'échantillon est dans un format brut, le programme vous demandera de donner une fréquence d'échantillonnage implicite (en Khz).

Lorsque vous êtes prêt à convertir les sons et à les sauvegarder sur la disquette,

sélectionnez l'option SAUVEGARDER du menu EDITER. La conversion peut prendre un peu de temps mais enfin c'est tout!

Utilisation de la mémoire

La quantité de mémoire dont le programme a besoin pour ranger les échantillons dépend de la taille courante du tampon du texte. Comme je l'ai mentionné auparavant, elle peut être définie au début du programme en utilisant la commande DEFINIR TAMPON. Si vous augmentez cete valeur, vous créez davantage d'espace pour la conversion. Mais attention! Le programme doit en fait contenir les échantillons et la nouvelle banque en mémoire. Pour calculer la mémoire utilisée par le programme de conversion, il faut multiplier la taille des échantillons par deux. Si cette taille est inférieure à la quantité de mémoire disponible, vous pouvez alors augmenter l'affectation sans danger. Note: Sur des machines à mémoire étendue, seule la mémoire RAPIDE peut être utilisée à cette fin.

Suggestions

Et si on demandait au programme de convertir les échantillons en format AMOS au fur et à mesure qu'il les charge ou si on ajoutait de nouvelles routines pour convertir les échantillons provenant d'autres ordinateurs tels que le Acorn Archimedes? N'oubliez-pas d'envoyer tous vos nouvelles créations améliorant le programme à Mandarin Software, ils intéresseront d'autres utilisateurs.

27 Les Jeux

Les Amostéroïdes

En l'an 2020, la Terre se trouvait dévastée, ravagée par la pollution et la sur-industrialisation. Vous offrez de piloter le dernier super vaisseau de combat Iconoclaste MKII. Malheureusement, pour résister aux champs de gravitation intenses, les savants ont dû installer des moteurs anti-grav consommant beaucoup d'énergie au détriment de l'armement. Votre seule défense (et moyens d'attaque) est une sorte de bouclier qui renverse le champs et attire son énergie des environs. Plus vous rencontrez d'astéroïdes en déplaçant votre vaisseau le plus lentement possible, plus votre force du bouclier sera grande. Malheureusement, la rapide installation des nouveaux moteurs (et du bouclier) ne vous laisse aucune place à bord de votre vaisseau de combat pour un interrupteur de secours et si votre bouclier est à court d'énergie, une réaction en chaîne se produira dans les moteurs anti-grav entraînant une implosion spectaculaire (et votre mort).

Les commandes

JOYSTICK A GAUCHE	Fait tourner votre vaisseau vers la gauche
JOYSTICK A DROITE	Fait tourner votre vaisseau vers la droite
JOYSTICK VERS LE HAUT	Active des propulseurs anti-grav
BOUTON TIR	Active les boucliers de renversement de champs
TOUCHE P	Pause/ reprise du jeu

Générique

Programmaton et conception: Peter J Hickman.
Musique: D J Nuttall.
Graphismes: Peter J Hickman, C D White avec assistance digitale d'Adam Fothergill.
Nos remerciements à Aaron Fothergill pour les trucs et astuces.

Notes techniques

Les Amostéroïdes se configurent eux-mêmes à la quantité de mémoire disponible, ce qui signifie malheureusement qu'avec un Amiga de 512k vous n'avez pas la musique ou les arrière-plans d'écran (bien que vous puissiez jouer avec eux séparément). Les arrière-plans d'écran sont remplacés par les arcs-en-ciel copper qui semblent presque aussi jolis que les images.

Notes générales

Les Amostéroïdes sont le résultat de nombreux mois de travail avec le langage d'animation AMAL (the **AM**os **A**nimation **L**anguage). Ils utilisent des chaînes longues et compliquées pour permettre au vaisseau et aux astéroïdes d'atteindre une vitesse rapide. Tout est géré à l'intérieur de l'AMAL; la partie Basic du programme produit seulement le bruit des explosions et actualise le score. Tous les registres externes AMAL sont utilisés en tant que système de passage de message à la partie Basic du programe afin qu'elle sache à quel moment un astéroïde a explosé (elle joue alors l'échantillon d'explosion).

Suggestions

Vous pouvez souhaiter améliorer le programme de Peter Hickman. Vous pourriez ajouter une option pour permettre à deux joueurs de jouer en même temps ou de créer des ennemis attaquant intelligemment; vous pouvez inventer quelques formes de système de bonus (peut-être permettre aux joueurs de remplir leur bouclier).

Amusez-vous bien et envoyez vos améliorations à Mandarin Software; cela nous ferait plaisir de les voir.

Castle AMOS

Cela se passe dans l'avenir. Vous avez terminé votre chef d'oeuvre AMOS mais il manque un peu de vitesse. La réponse se trouve dans le Compileur AMOS qui est sorti dernièrement.

Vous enfilez vos bottes de pluie et votre imper (ouai! temps typiquement britannique), vous vous dépêchez d'attraper le métro pour vous rendre à votre boutique informatique préférée. Vous sortez aux Halles et vous vous ruez dehors; en regardant autour de vous, vous vous apercevez que toute la place et la station de métro ont disparu (vous êtes pourtant un bon observateur!)

A ce moment-là, vous vous rendez compte qu'une ou deux choses se sont produites: vous n'avez pas plu au contrôleur ou un court circuit dans le nouveau portillon d'accès a ouvert une porte inter-dimensionnelle et vous a transporté en Transylvanie.

Eh bien, une carte d'une journée de métro ne va certainement pas vous donner le compilateur en Transylvanie - peut-être que la réponse se trouve dans ce château au loin...

Les commandes

A la différence de nombreuses aventures, Castle AMOS vous permet de contrôler presque tout le jeu avec la souris. Vous pouvez sélectionner toutes les commandes à partir d'une liste défilante contenue dans le défilement de droite.

Lorsque vous vous déplacez vers un nouveau lieu ou utilisez la commande PICTURE pour visualiser l'emplacement courant, les défilements disparaîtront très vite de l'écran. En cliquant sur le bouton de la souris, vous pourrez les revisualiser. C'est à peu près tout ce que vous devez savoir, après tout je ne veux que ce soit trop simple!

Générique

Programmation: Abdul M Kalim et Peter J Hickman.
Conception: Peter J Hickman et C D White.
Musique: D J Nuttall.
Graphismes: C D White et Peter J Hickman.
Nos remerciements à Richard Vanner (salut Comte Dick!) et François pour nous laisser mutiler leurs images!!

Magic Forest

C'est l'automne dans la Forêt Magique et les fruits commencent à tomber des arbres. C'est le travail de Wilf de ramasser les pommes avant qu'elles touchent le sol et qu'elles soient abîmées. Cependant, les mauvais Broyeurs de pommes au visage vert se sont

déplacés sur les plate-formes de ramassage et poussent les pommes sur le sol!
Vous contrôlez Wilf et devez ramasser votre quota de pommes par niveau. Les pommes rouges valent 70 points (une pomme en moins de votre quota), les cerises rouges valent 100 points (deux pommes en moins de votre quota) et les pommes bleues valent 200 points (trois pommes en moins de votre quota).

Si vous laissez tomber trop de pommes, ramassez une pomme empoisonnée ou touchez une pomme des Broyeurs de pommes, vous perdrez un de vos trois vies. Les Broyeurs de pommes ne cessent de chanter une incantation ce qui appelle bientôt un orage. Vous ne le savez peut- être pas mais rester sous un arbre pendant un orage est très dangereux et cela n'est pas recommandé à ceux qui ne tolèrent pas les chocs électriques. Si vous vous attardez trop longtemps, il y a de grandes chances que la foudre ajoute un peu plus de vitesse à votre course.

Les commandes

Utilisez un joystick dans le port 1 (et non pas le port de la souris) pour déplacer Wilf.
Poussez le joystick vers la gauche pour le diriger vers la gauche.
Poussez le joystick vers la droite pour le diriger vers la droite.
Pour faire sauter Wilf, poussez le joystick vers le haut.

Suggestions

Cette version de Magic Forest comprend cinq niveaux. L'auteur va peut- être écrire une version améliorée dans l'avenir. Vous pourriez peut- être utiliser AMOS TAME pour créer d'autres niveaux. Envoyez-les à Mandarin Software et nous serons ravis de les voir.

Générique

Programmation et conception du jeu: Aaron Fothergill
Graphisme et idée de jeu: Adam Fothergill
Musique: Aaron et Adam Fothergill
SOS de toute dernière minute: Richard Vanner (Les propriétaires du A500 viennent juste d'avoir ce jeu! Bravo Aaron!)

Number Leap

Fosdyke la grenouille n'était pas très heureuse; elle avait des problèmes à apprendre ses tables de multiplication à l'école. Cela la rendait très triste: elle faisait quelque fois semblant d'être malade pour ne pas avoir à se lever mais un jour, son cousin Arthur, le comptable amphibie lui proposa une nouvelle façon amusante de les apprendre.
Arthur creusa un bassin, planta des nénuphars et numérota chacun d'eux. Et Fosdyke n'a alors plus qu'à choisir une table qu'elle veut pratiquer et sauter sur les nombres qui apparaissent dans la table en question. Si elle se pose sur un nénuphar qui n'est pas dans la table, elle coule. Si elle a juste, elle traverse le bassin en sautant de joie.

Les commandes

Pour contrôler Fosdyke, vous pouvez utiliser soit un joystick que vous brancherez dans le port 1 (et non le port de la souris!), soit les touches du curseur. Pour sélectionner une

table de multiplication, il vous suffit de déplacer Fosdyke sur le nénuphar approprié et d'appuyer sur tir (ou la barre d'espacement si vous utilisez les touches du curseur). Au cours du jeu, vous pouvez appuyer sur une touche (sauf les touches du curseur) pour entendre l'ordinateur dire le résultat figurant sur le nénuphar où se trouve Fosdyke.

Générique

Programmation: Peter J Hickman.
Graphismes: C D White et Peter J Hickman.

28 Bibliothèque du domaine public

Adresse

AMOS PD LIBRARY, c/o Sandra Sharkey,
25 Park Road, Wigan,
Lancashire,WN6 7AA
Angleterre

Fonctionnemen

La bibliothèque AMOS DP est une bibliothèque du spécialiste comprenant des programmes et des utilitaires jugés dignes d'intérêt aux utilisateurs AMOS.
Tous les programmes comprennent un code dévérouillé pour que vous puissiez les examiner et apprendre à réussir certains effets.

Un catalogue listant tous les programmes courants est disponible. Si vous souhaitez obtenir un exemplaire, envoyez une enveloppe timbrée adressée à votre nom et un coupon-réponse international à l'adresse ci-dessus. Aucune cotisation n'est demandée pour utiliser la bibliothèque qui est ouverte à tout le monde.

Toutes les disquettes de la bibliothèque sont amorçables. Si ce sont des programmes AMOS déroulables, elles se dérouleront ou dans le cas d'une disquette pleine, un sélecteur ou quelque chose de similaire s'affichera. A l'amorçage, elles affichent le logo Bibliothèque AMOS DP, faisant apparaître les coordonnées de la biblothèque. Plus de recherche de ce petit bout de papier!

Dans la mesure du possible, toutes les disquettes utilisent les toutes dernières versions des programmes, y compris le module de Musique et RAMOS. Elles utilisent les blocs d'amorçage pour contenter VirusX.

Veuillez noter que la Bibliothèque AMOS PD ne fait pas partie du Club AMOS et ne peut pas répondres aux demandes du Club AMOS.

Paiements

Les paiements en provenance de l'étranger doivent être effectués en livres sterling, soit par Mandat International, Eurochèque ou Mandat Postal. Nous ne pouvons pas accepter les paiements en d'autres devises ou les paiements par carte de crédit. Les paiements en timbres postaux britanniques sont possibles et permettent d'économiser les commissions demandées par les mandats postaux.

Les chèques et mandats postaux doivent être barrés, libellés à l'ordre de AMOS Public Domain Library et envoyés à l'adresse ci-dessus

Les disquettes APD coûtent 2,50 livres sterling pour le Royaume-Uni, 2,75 livres sterling pour l'Europe et 3,00 pour le reste du monde. Les prix comprennent les frais

d'envoi et d'emballage.

Si vous nous fournissez vos propres disquettes vierges, déduisez 1 livre sterling par disquette.

Si vous achetez trois disquettes ou plus, déduisez alors 20 pence par disquette depuis la première achetée, ce qui vous permet d'économiser au moins 60 pence.

Directives à suivre lors de la soumission des disquettes

Lorsque vous avez fini d'écrire votre super-production, vous pouvez l'envoyer et nous l'intégrèront peut-être à la bibliothèque. Voici quelques conseils à suivre pour que nous gagnions tout deux du temps:

Votre envoi doit comprendre une documentation, si possible sur la disquette contenant votre programme, et n'oubliez pas de spécifier si le programme demande une mémoire supérieure à 512k.

Vérifiez toujours que vos disquettes sont exemptes de virus avant de les soumettre et n'oubliez pas de choisir une disquette du catalogue en échange de la votre. Si vous désirez une des disquettes sous licence, vous devez nous envoyer les droits d'auteur que nous expédierons à ce dernier.

Les programmes sous licence

Les disquettes de cette section coûtent un peu plus que le prix DP ordinaire car des droits sont versés aux auteurs de ces programmes. Ils ne sont pas du domaine publique et la distribution non autorisée de ces disquettes n'est pas permise.

LPD1: *Livre de coloriage (Colouring Book)* - par Trevor Prince. Un jeu simple pour les enfants depuis la maternelle. Chargez une des six images qui vous sont proposées et une comptine. Cliquez sur la couleur que vous voulez et coloriez l'image. Une nouvelle version est en développement comprenant un dessin à colorier, de l'animation et bien d'autres possibilités. Il se déroule sur des machines d'´ Mo.

3,50 livres sterling pour le Royaume-Uni, 3,75 livres sterling pour l'Europe et 4 livres sterling pour le reste du monde.

LPD4: *THINGAMAJIG par Len Tucker* - Est-ce-que vos enfants (ou vous- même) aiment les puzzles? Si vous répondez par "oui", c'est le programme qu'il vous faut. La disquette comprend 24 images. Deux options vous sont proposées, FACILE ou DIFFICILE - et c'est dur, croyez-moi! L'image de votre choix vous est présentée et vous devez assembler les morceaux correctement. Il y a une fonction SOS si vous séchez. Un programme très professionnel et très bien présenté. Il se déroule sur des machines d'´ Mo.

3,50 livres sterling pour le Royaume-Uni, 3,75 livres sterling pour l'Europe et 4 livres sterling pour le reste du monde.

LPD5: *JUNGLE BUNGLE par Len Tucker* - 1 Mo seulement. Une aventure conduite par icône, écrite pour les enfants mais assez captivante pour l'aventurier le plus endurci. De beaux graphismes, un son échantillonné et de l'animation. Attraper des gouttes de pluie

pour collecter de l'eau est amusant et guettez le singe qui vole des bananes!

3,50 livres sterling pour le Royaume-Uni, 3,75 livres sterling pour l'Europe et 4 livres sterling pour le reste du monde.

LPD7: *4 WAY LYNX par Andreas Andreou* - 1 Mo et un joystick nécessaires. Documentation complète sur disquette. Le but de ce puzzle à 22 niveaux est de relier un nombre spécifique de tuiles à la tuile principale se trouvant au centre. Un certain nombre d'objectifs doit être atteint à chaque niveau avant de pouvoir avancer au niveau suivant. Vous pouvez même créer vos propres niveaux au nombre de 40 si vous le voulez. Les niveaux deviennent de plus en plus difficiles au fur et à mesure que vous avancez. Jeu très passionnant!

3,50 livres sterling pour le Royaume-Uni, 3,75 livres sterling pour l'Europe et 4 livres sterling pour le reste du monde.

Les programmes disponibles

La principale collection AMOS DP a été divisée en 6 catégories pour que vous puissiez trouver plus facilement les disquettes qui vous intéressent le plus. Nous avons sélectionné ci-dessous quelques disquettes du catalogue mais plus de 100 programmes vous sont proposés.

Les disquettes utilitaires

Certains de ces utilitaires proviennent du DP général alors que d'autres ont été écrits en AMOS. Tous sont jugés dignes d'intérêt aux programmeurs AMOS.

APD1: *GAMES MUSIC CREATOR* - un utilitaire puissant et facile à utiliser vous permettant de créer des mélodies. Il utilise tous les quatre canaux et il est livré avec des fichiers DOC. Servez-vous du programme de convertion GMC-to-AMOS pour modifier un "bloc" de données musicales et jouez-les en AMOS au moyen de la simple commande MUSIC.

APD6: *UTILITAIRE DE CONVERSION STOS EN AMOS* - Programme permettant la conversion des fichiers IBM et Atari ST. Il comprend un convertisseur IFF et d'autres fichiers de conversion AMOS STOS.

APD7: *VIRUSX ET AUTRES UTILITAIRES* - D'autres utilitaires ont été ajoutés à cette disquette, y compris le VIRUSX4.1. Un complément utile à la collection de tous les utilisateurs d'AMOS.

APD31: *CONCEPTEUR D'ECRAN (SCREEN DESIGNER) par James Robert Crosby* - Le but de ce programme est de permettre aux utilisateurs AMOS de créer de grands écrans IFF qui peuvent être chargés dans n'importe quel de leurs propres programmes. Cette disquette comprend également un grand nombre d'autres routines et d'utilitaires se chargeant en AMOS.

APD76: *RAINBOW WARRIOR par Martyn Brown* - Documentation complète sur la disquette. RW vous donne l'occasion de "peindre" en utilisant le copper et les arcs-en-ciel et de sauvegarder vos créations pour les intégrer dans vos propres programmes. Le programme sort des blocs ascii afin de pouvoir les fusionner directement! RW est écrit en AMOS pour que les utilisateurs puissent y adapter, mélanger et fusionner leurs propres compositions; le programme sort également des données pour K-Seka, Devpac, Binary, Decimal, Hex et ses propres formats. La disquette comprend également toutes les dernières versions des meilleurs utilitaires AMOS, c'est-à-dire les tout derniers convertisseurs de soundtracker, squash-a-bob, le tout dernier RAMOS. Il y a même une base de données "BOMBASE" réalisée par Gareth Lancaster. Cette disquette est un must pour tous les utilisateurs sérieux.

APD83: *AMOS PAINT par Jounii Poullnen* - Il fonctionne avec ´ Mo mais une mémoire d'1Mo ou plus est recommandée. Il existe 2 versions d'AMOS-Paint, PAL et NTSC. Editez la séquence de mise en route de la disquette AMOS Paint pour permettre au programme d'amorcer, puisqu'elle est une version PAL par défaut. Les fonctions: 2 à 64 couleurs (vous pouvez charger les images HAM), interface-utilisateur conviviale avec cache-menu automatique, peinture sur l'intégralité de l'écran, Auto-Scrolling lisse en temps réel sur des super bitmaps peuvent sauvegarder et charger les fichiers de format IFF et SPACK, permet l'itération de couleurs en utilisant SHIFT UP ou SHIFT DOWN, offre l'utilisation de tous les outils normaux de peinture, le mode de styles de peinture.

Les jeux et les programmes AMOS

Une des façons les plus faciles d'apprendre à programmer est de voir comment une autre personne s'y est prise et avec les disquettes AMOS DP, tout est possible.

APD2: *LA CHASSE AU TRESOR (TREASURE SEARCH) par Peter Hickman* - Cherchez le trésor dans ce jeu éducatif intelligent en vous servant des coordonnées. Les graphismes et le son d'excellente qualité amusent tout le monde de 2 à 90 ans!

APD1: WORD SQUARE SOLVER + JEUX - Le jeu des lettres est facilité avec ce programme. Tapez simplement les lettres et laissez l'ordinateur trouver les mots pour vous. Les jeux suivants se trouvent également sur la disquette: Demolition Mission, Grub Grabber, Space Invaders, Pacman et un jeu du sous-marin.

APD32: *PROGRAMMES AMOS 1 par Gary Fearn et Nadeem* - Comprend CLI.AMOS, Myfirstdemo.AMOS, Spritestars.AMOS et Textview.AMOS par Gary. Une démo réalisée par Nadeem se trouve également sur la disquette. Beaucoup de code source que vous pouvez étudier et avec lequel vous pouvez jouer.

APD54: *PROGRAMMES AMOS 2 par Gary Shilvock et Jason Anthony* - Cette disquette comprend une compilation d'utilitaires, de routines et de démos. Encore plus de code source à examiner et à adapter.

APD62: *ARCADIA* - Retirez le 2ème lecteur sur ´Mo. C'est mon jeu préféré. Excellente jouabilité, version passionnante du célèbre jeu Arkanoid/Breakout. Vous avez même un concepteur de niveau pour pouvoir éditer ce jeu tout votre content. Très recommandé.

APD85: *REVERSI & AMOS SNAKES* - Reversi est une version super fantastique d'Othello. Je ne peux pas m'arrêter de jouer. Jouez avec l'ordinateur ou un ami. Une excellente version du jeu de l'oie. Mes gosses se battent pour la première place alors que je suis toujours en train de lancer un six!! Exactement comme le jeu de société. Je n'ai pas de chance. Deux jeux excellents sur un seul disque. Très recommandé. Se déroulent sur des machines d'´ Mo.

APD97: *DYNAMITE DICK par Adam Leech* - Un petit jeu tout à fait sympa dans lequel vous ramassez un trésor, tuez les monstres et trouvez la clé pour monter au niveau suivant. Egalement sur la disquette par le même programmeur se trouvent une série de diapos et une démo. Recommandé.

Les démos

Les démos peuvent constituer une excellente base à de bonnes routines et elles vous sont toutes accessibles, utilisateur AMOS veinard.

APD9: *AMOS BIG DEMO V4 par Peter Hickman* - La démo écrite par Peter Hickman pour mettre AMOS en valeur. Bien qu'elle fût d'abord écrite sur la base d'une version précoce d'AMOS, elle est toujours impressionnante. La disquette comprend un extrait de Newsflash, interview entre Martyn Brown et François Lionet, intéressant.

APD22: *L'ECOLE DES MALINS 3, démo par Peter Hickman* - La troisième série de l'Ecole des Malins de Database Software. Installez-vous bien et laissez nounours vous présenter les programmes des trois fourchettes d'âge de ce nouveau progiciel. Il vous montre combien AMOS peut être utile sous la forme de démo pour promouvoir des logiciels.

APD81: *JUKEBOX DEMO* - Il vous faut le APD82. Deux lecteurs de disques musicaux. C'est le programme qui figure dans les pages publicitaires d'AMOS de Mandarin Software.

APD82: *JUKEBOX DEMO* - Il vous faut le APD81. 2ème disquette complétant celui ci-dessus.

APD99: *DEMO BENSON 1* par Leslie Benzies - La démo gagnante dans un concours parrainé par Mandarin. Comprend un code source excellent pour le scrolling, etc... Recommandé.

APD100: *AMOS BIG DEMO II par Peter Hickman* - Cette démo fut utilisée par Mandarin à des salons informatiques pour promouvoir AMOS. Comprend encore plus d'excellent code source à examiner et à étudier. Très recommandé.

Disquettes Art et Polices de caractère

PD78: *IFF Picture Disc #1* - Elle comprend 14 polices y compris TIMES, BOOKMAN, HELVETIC et TINY.

APD4: *FONTS DISC #2* - 13 polices y compris AVANT GARDE, CELTIC, PALO ALTO, BASEL, PEIGNOT et ALDUOUS.

APD5: *FONTS DISC #3* - 14 polices y compris BROADWAY, CAMELOT, FUTURE, STENCIL et VANCOUVER.

APD38: *IFF FONTS DISC* - Un choix de polices IFF converties du ST par "The Skunk" ainsi que d'autres polices AMIGA supérieures par Spadge. Celles-ci peuvent être coupées en bobs et utilisées avec AMOS. Toutes les polices sont affichées dans une série de diapos accompagnées de musique.

La musique AMOS

Toutes ces disquettes sont amorçables et comprennent un lecteur de musique en AMOS pour que les utilisateurs puissent facilement écouter la musique. Voir la collection australienne et Music General qui vous offrent d'autres disquettes d'échantillons, des Modules ST et des Instruments.

Il existe bien d'autres disquettes que nous pouvons énumérer dans le manuel mais vous les trouverez toutes dans le catalogue de la bibliothèque AMOS DP.

Music general

Nous avons un choix de modules de Soundtracker, des Instruments et des Echantillons que vous pouvez utiliser avec le GMC, etc... Vous trouverez tous les renseignements à ce sujet dans le catalogue.

Tous les échantillons ont été laissés dans le format IFF puisqu'ils peuvent être utilisés dans AMOS au moyen de l'utilitaire de création d'échantillons. Les disquettes d'échantillons sont toutes amorçables, affichent un sommaire et permettent à l'utilisateur de les écouter en utilisant un petit utilitaire.

Les disquettes Instrument sont destinées à être utilisées avec le GMC et la série SOUND/NOISE/PROTRACER et utilisent donc le système conventionnel d'étiquetage "ST-". Chacun comprend une liste pré- définie et une méthode d'écoute de tous les échantillons par Perfect Sound.

Les disquettes des modules sont toutes amorçables et fournissent des informations. Elles permettent à l'utilisateur d'écouter les modules en utilisant le progiciel Intuitracker DP. Elles ont été également converties en fichiers ABK et les modules figurent sur les disquettes Music.ABK.

29 Explication des messages d'erreur

Messages d'erreur de l'éditeur

Lorsqu'une erreur se produit, un des messages suivants apparaît à l'écran lors de l'édition de votre programme.

Aucune erreur: Aucune erreur n'a été interceptée lors de la procédure de test.
Erreur de syntaxe: La syntaxe (grammaire) de la ligne courante n'est pas correcte. Recherchez la bonne syntaxe dans ce manuel ou sur la fiche de référence.
Fin de texte: Le curseur est arrivé à la dernière ligne de votre programme courant.
Haut du texte: Le curseur du texte a atteint le haut de votre programme. L'appui sur la flèche ascendante n'aura aucun effet.
Impossibilité d'entrer le programme dans le tampon de l'éditeur: Vous obtenez ce message lorsque l'espace nécessaire au chargement d'un programme est plus important que la zone disponible dans le tampon de l'éditeur. En répondant par oui à la question posée par AMOS, le tampon du texte prendra exactement la taille du programme devant être chargé. Vous ne pourrez pas développer votre programme mais en réduire seulement sa taille. Si cela est nécessaire, vous pouvez appeler S.BUFFER du menu cherche pour décomprimer le tampon du texte. En sélectionnant non, vous suspendez le chargement du programme et l'espace minimum nécessaire du tampon s'affiche sur la ligne d'information.
Introuvable: La commande cherche précédente n'a pas abouti.
Ligne trop longue: L'éditeur d'AMOS peut seulement traiter des lignes contenant au plus 255 caractères.
Marque non définie: Vous ne pouvez pas vous déplacer vers une marque parce que vous n'avez pas défini de bornes.
Mémoire pleine: Il n'y a plus de mémoire disponible pour contenir vos programmes. Essayez d'utiliser CLOSE WORKBENCH pour reprendre les 40k utilisés par l'écran Workbench de l'Amiga.
Pas une procédure: PLIER/DEPLIER fonctionne seulement si vous avez positionné le curseur du texte sur une procédure.
Quel bloc?: Vous ne pouvez COUPER/COLLER un bloc avant de l'avoir défini.
Tampon de texte plein: Vous êtes à court d'espace dans la zone de l'éditeur. Sauvegardez votre programme sur le disque et augmenter le tampon en utilisant S.BUFFER (dans le menu CHERCHE). Si vous avez toujours des difficultés, divisez votre programme en plusieurs parties et exécutez- les l'une après l'autre en utilisant la commande RUN de l'AMOS BASIC.
Trop de variables en mode direct: Il n'y a pas assez d'espace dans la table des variables pour contenir vos variables en mode direct. Le mode direct vous permet de créer au plus 64 nouvelles variables mais ce nombre peut être restreint si votre programme prend trop de mémoire.

Tampon des nom de variable trop petit: AMOS a un tampon qui contient les noms de toutes les variables. Si vous entrez un nom trop long, vous produisez une erreur. Vous pouvez changer la taille initiale du tampon depuis l'accessoire CONFIG.

Les erreurs de programme

Chaque fois que vous déroulez un de vos programmes ou le vérifiez au moyen de la commande TEST de la fenêtre du MENU, AMOS effectue un test complet de toutes les instructions. Ceci vous permet de vous débarasser de la plupart des erreurs directement depuis l'éditeur sans être gêné par l'exécution de vos programmes.
Voici une liste complète de ces messages d'erreur:
Cette variable est déjà définie entant que SHARED: Vous ne pouvez pas définir la même variable plus d'une fois dans une seule procédure.
Ce tableau n'est pas défini dans le programme principal: Vous avez essayé d'accéder à un tableau dans une procédure qui n'a pas été dimensionné dans le programme principal.
DO sans LOOP: Les instructions DO et LOOP sont inséparables. Chaque instruction DO doit être accompagnée d'une seule commande LOOP.
ELSE sans ENDIF: Vous avez omis la commande ENDIF finale d'un test structuré IF.
ELSE sans IF: La commande ELSE peut seulement être utilisée dans un test structuré.
END IF sans IF: L'instruction ELSE a été trouvée dans votre programme et elle ne correspond pas à une instruction IF appropriée.
Erreur de syntaxe: La syntaxe (grammaire) de la ligne courante est incorrecte. Cherchez la bonne syntaxe dans le manuel.
Etiquette déjà définie: Chaque étiquette ou procédure ne peut être définie qu'une seule fois dans votre programme.
Etiquette non définie: AMOS ne peut pas trouver l'étiquette que vous avez spécifiée dans l'instruction.
Extension non chargée: Vous avez essayé de faire fonctionner un programme en intégrant une des nouvelles commandes proposées par le fichier d'extension. Vérifiez que les extensions appropriées ont été installées sur votre disque d'initialisation et que les extensions sélectionnées peuvent être utilisées depuis l'accessoire CONFIG.
FOR sans le NEXT correspondant: Dans le programme, une commande FOR n'est pas suivie de l'instruction NEXT attendue.
IF sans ENDIF: Les instructions à l'intérieur d'un test structuré IF doivent toujours être suivies d'une seule instruction ENDIF. Ne confondez pas ces tests avec la commande IF...THEN; elles sont complètement différentes.
Instruction valide uniquement dans le procédure: La commande SHARED peut seulement être utilisée A L'INTERIEUR d'une définition de procédure.
Les instructions DATA doivent commencer au début d'une ligne: Les instructions DATA de votre programme doivent être placées au tout début d'une ligne (à l'exception des définition de LABEL).
Les limites de la procédure doivent être isolées: Les instructions PROCEDURE et END PROC doivent figurer sur une ligne à part.
LOOP sans DO: Une commande LOOP a été détectée et elle n'est pas associée de l'instruction DO.
Mauvaise structure: Toutes les boucles emboîtées doivent être placées l'une dans

l'autre dans LEUR INTEGRALITE. Il est interdit que les boucles "se croisent". Exemple:

```
Do
  If A=B
Loop
  Print A
Endif
```

Ce **n'**est **pas** autorisé.

NEXT sans FOR: AMOS a rencontré une instruction NEXT qui n'est pas associée à une commande précédente FOR.
Nombre de paramètres incorrect: Vous avez essayé d'entrer un nombre incorrect de valeurs dans une instruction ou une procédure.
Nombre insuffisant de boucles: Le nombre de boucles que vous avez spécifié dans une commande EXIT ou EXIT IF est supérieur au nombre de boucles actives.
Pas de THEN dans un test structuré: La commande IF...THEN NE PEUT PAS être placée dans un test structuré. Utilisez plutôt IF...ENDIF.
Pas de saut au beau milieu d'une boucle!: Vous ne pouvez pas vous brancher directement à l'intérieur d'une boucle en utilisant une instruction GOTO ou GOSUB. Toutefois, une fois que vous vous trouvez dans une boucle, vous pouvez en sortir.
Procédure non fermée: L'instruction END PROC manque dans l'une de vos procédures.
Procédure non définie: La procédure que vous avez appelée n'existe pas actuellement dans votre programme.
Procédure non ouverte: Une instruction END PROC a été détectée sans définition de PROCEDURE correspondante.
REPEAT sans le UNTIL correspondant: Une instruction REPEAT existe dans votre programme, mais elle n'est pas suivie de l'instruction UNTIL.
Shared doit être seul sur la ligne: La commande SHARED doit être la seule instruction de la ligne courante.
SET BUFFER doit être la première instruction du programme!: La commande SET BUFFER doit toujours être la première ligne de votre programme, après les Rems.
Tableau déjà dimensionné: Il est seulement possible de dimensionner le même tableau lorsque vous vous trouvez dans votre programme Basic.
Tableau non dimensionné: L'élément que vous spécifiez dans l'expression n'appartient à un tableau préalablement dimensionné.
Tampon de variable trop petit: Lors de l'essai d'un programme, AMOS utilise une zone de mémoire réservée pour les variables. Si la zone de ce tampon déborde, vous obtiendrez ce message d'erreur. Utilisez la commande SET BUFFER pour demander davantage de mémoire pour vos espaces variable (si la mémoire le permet).
UNTIL sans REPEAT: La commande UNTIL n'est pas associée à l'instruction précédente REPEAT dans votre programme.
Utilisez des parenthèse vides pour désigner un tableau: Pour définir un tableau comme SHARED, utilisez une commande comme suit:

```
Shared Array()
```

Il est interdit d'ajouter les dimensions du tableau.
WEND sans WHILE: L'instruction WEND n'est pas associée à une commande WEND précédente dans votre programme.
WHILE sans le WEND correspondant: AMOS ne peut pas trouver une instruction WEND correspondante pour l'associer au WHILE courant.

Message d'erreur d'exploitation

Les messages suivants sont produits lorsque le programme AMOS rencontre une erreur lors de son déroulement. AMOS suspendra alors le programme et mettra l'instruction courante en surbrillance. Au retour à l'éditeur, le curseur se placera immédiatement sur la ligne où l'erreur se trouve.

Si vous utilisez un programme d'interception de l'erreur, vous pouvez souhaiter que le message soit associé à un numéro particulier. Tapez donc une ligne comme suit:

```
Error Errn
```

Chaque erreur *d'exploitation* a son propre numéro d'erreur qui est cité ci-dessous.
Une commande FOLLOW spéciale peut également être utilisée pour *voir* comment vos variables changent lors du déroulement de votre programme. Les commandes FOLLOW sont présentées ci-dessous:

FOLLOW *(Visualiser l'état d'une ou plusieurs variables)*

FOLLOW [liste d'expressions...]

La commande suivante arrête l'exécution du programme jusqu'à ce que l'utilisateur appuie sur la barre d'espacement. Si les expressions sont citées, alors leurs résultats seront affichées dans une fenêtre indépendante. Cette fenêtre est identique à celle en mode direct en ce sens qu'elle ne perturbe pas l'écran de votre programme. Vous pouvez monter ou descendre l'image-écran en utilisant les flèches ascendante ou descendante.

Le système ne suit pas les procédures *pliées*, mais vous permet de suivre la procédure ou la routine qui doit être inspectée.

FOLLOW OFF *(Arrêter les commandes Follow précédentes)*

FOLLOW OFF

Cette commande efface les instructions FOLLOW précédentes et fait disparaître la fenêtre FOLLOW de l'image-écran.

AMAL: AUtotest déjà ouvert (111): Un autotest AMAL a été défini à l'intérieur d'une autre commande autotest. Ce n'est bien sûr pas permis.
AMAL: chaîne d'animation trop longue (113): La longueur du programme AMAL courant est supérieure au maximum de 65536 octets. Essayez de diviser votre programme en plus petites unités. Il est tout à fait possible d'animer le même objet en utilisant plusieurs canaux AMAL.

AMAL: erreur de syntaxe (107): Il y a eu une erreur dans la séquence d'animation que vous avez spécifiée en utilisant la commande Anim. Recherchez les fautes de frappe. Il est très facile d'entrer un point "." au lieu d'une virgule "," par inadvertance.
AMAL: étiquette déjà définie (114): Ce message d'erreur se produit lorsque vous essayez de vous brancher sur une étiquette qui n'existe pas dans la chaîne d'animation AMAL.
AMAL: etiquette non définie: (109): AMOS a rencontré deux versions de la même définition d'étiquette de votre programme AMAL. Rappelez-vous que toutes les étiquettes comprennent seulement une lettre majuscule.
AMAL: Next sans For (108): Ce message indique qu'il existe une erreur dans une de vos chaînes d'animation AMAL. Chaque commande Next doit s'associer à une seule instruction For. Vérifiez si c'est bien le cas dans tous les commentaires de vos programmes AMAL.
AMAL: instruction valide seulement dans un Autotest (112): Vous pouvez seulement utiliser les commandes Direct ou eXit dans une commande AMAL AUTOTEST.
AMAL: saut vers/dans un Autotest (110): Il est interdit de se brancher directement dans une commande AUTOTEST du programme AMAL principal. Pour sortir d'AUTOTEST, utilisez les commandes eXit ou Direct.
Appel illégal de fonction (23): Ce message apparaît si vous commettez une erreur en entrant des valeurs dans une commande AMOS. Recherchez la liste complète des paramètres autorisés dans le manuel d'utilisation.
Aucune zone définie (73): Avant d'utiliser SET ZONE, vous devez d'abord allouer de la mémoire à RESERVE ZONE.
Banque déjà réservée (35): Vous avez essayé de créer une banque mémoire qui existe déjà. Notez que les banques de 1 à 4 sont normalement utilisées pour contenir respectivement vos sprites, vos icônes, la musique et les définitions d'icônes.
Banque non réservée (36): Ce message d'erreur indique que la banque que vous avez sélectionnée n'a pas été créée en utilisant RESERVE. Il peut également se produire après l'exécution de commandes comme PASTE ICON ou SAMPLAY qui chargent automatiquement des informations d'une banque spécifique de la mémoire.
Bloc introuvable (65): Le bloc que vous avez spécifié dans cette instruction n'a pas été créé à l'aide de GET BLOCK.
Bob non défini (68): Le bob que vous avez essayé de manipuler n'a pas été défini au moyen de la commande précédente Bob. Vous pouvez obtenir ce message d'erreur après avoir commis une erreur dans une instruction PASTE BOB.
Ce n'est pas une disquette AmigaDOS (92): A moins que vous utilisiez un programme tel que CROSSDOS, AMOS peut seulement lire les disquettes qui ont été créées sur l'Amiga. Par conséquent, les disquettes pour PC et ST produiront ce message d'erreur.
Cette fenêtre n'a pas de bordure (63): Vous avez utilisé la commande BORDER dans une fenêtre qui n'a pas de contour.
Chaîne trop longue (21): Une chaîne a dépassé les 65000 caractères, maximum permis sous AMOS Basic.
Copper non désactivé (76): Vous avez essayé d'utiliser les commandes COP MOVE ou COP SWAP sans avoir d'abord désactivé la liste de copper normale à l'aide de COPPER OFF.
Dépassement de capacité (29): Un calcul a donné un résultat dépassant la taille maximale d'une variable.

Disque plein (88): Il n'y a plus de place sur le disque courant pour contenir vos données.
Disque protégé en écriture (84): AMOS ne peut pas sauvegarder des informations sur le disque courant parce qu'il est physiquement protégé. Glissez donc l'onglet de protection à l'écriture ou utilisez une autre disquette.
Disquette non valide (83): Lorsque vous insérez une disquette dans le lecteur, l'Amiga vérifie automatiquement sa validité. Si une erreur se produit au cours de ce procédé, vous obtiendrez ce message d'erreur. En raison de la complexité du système disque de l'Amiga, des erreurs de validation sont quelques fois générées. Pour résoudre le problème, utilisez le programme DISC DOCTOR de la disquette Workbench normale. Ce message d'erreur peut également se produire (cela nous est arrivé) si vous débranchez la prise et donc invalidé le disque dur!
Division par zéro (20): Vous avez essayé de diviser un nombre par zéro. Ce n'est pas autorisé en AMOS Basic ou dans tous les langages Basic.
Ecran déjà en DOUBLE BUFFER (69): Vous avez essayé d'appeler deux fois DOUBLE BUFFER au même écran.
Ecran non dual playfield (71): Vous pouvez seulement utiliser DUAL PRIORITY après avoir créé un double champ de lecture.
Ecran non ouvert (47): L'écran auquel vous avez essayé d'accéder n'a pas été préalablement ouvert au moyen de la commande SCREEN.
Entrée trop longue (99): Une chaîne d'entrée est trop longue pour une variable préalablement dimensionnée ou bien vous avez essayé d'entrer une ligne de plus de 1000 caractères au moyen de INPUT#.
Erreur dans le déclaration du Shift (53): Vous avez commis une erreur dans la séquence de couleurs utilisée dans les instructions SHIFT UP ou SHIFT DOWN.
Erreur de sprite (105): Les valeurs que vous avez entrées dans une commande SPRITE ne sont pas comprises dans les limites autorisées.
Erreur d'entrée/sortie (94): Il se peut qu'un de vos fichiers se soit altéré. Il n'est donc pas proprement accessible. Si le problème persiste, vérifiez soigneusement les connexions de l'unité de votre disque. Il vous faut peut-être réparer votre disque en utilisant le programme DISC DOCTOR se trouvant sur le disque Workbench initial.
Erreur dans le déclaration du Flash (52): Vous avez commis une erreur dans la chaîne d'animation utilisée pour définir une séquence de couleurs avec FLASH.
Erreur d'adresse (25): Ce message se produira si une adresse utilisée dans une commande DOKE, DEEK, LOKE ou LEEK a une valeur impaire.
Erreur non "RESUME" (3): Vous êtes sorti d'une routine de gestion de l'erreur sans corriger l'erreur en utilisant RESUME.
Fenêtres à contour pas au bord d'un (59): Vous ne pouvez pas positionner le cadre d'une fenêtre sur le bord de l'écran. Vous devez laisser au moins huit pixels entre le cadre de la fenêtre et le bord de l'écran afin de prévoir de l'espace pour la frontière de l'écran.
Fenêtre déjà ouverte (55): Vous avez essayé d'ouvrir une fenêtre qui est déjà ouverte.
Fenêtre non ouverte (54): Vous avez essayé d'accéder à une fenêtre qui n'existe pas.
Fenêtre trop large (57): La fenêtre demandée ne peut pas être ouverte parce qu'elle est trop grande pour entrer dans l'écran courant.
Fenêtre trop petite (56): La fenêtre demandée est trop petite. La taille minimale de la fenêtre est de 3 sur 3.

Fichier déjà existant (79): Il n'est pas possible de RENOMMER un fichier ou un répertoire dont le nom existe déjà sur le disque.
Fichier déjà ouvert (96): Votre accès à un fichier à l'aide de la commande OPEN ou APPEND n'a pas abouti parce que le fichier est déjà ouvert.
Fichier introuvable (81): Vous avez essayé d'accéder à un fichier ou à un répertoire qui n'existe pas dans le répertoire courant.
Fichier non ouvert (97): Votre programme a essayé de transférer des données vers ou en provenance d'un fichier qui n'a pas été préalablement ouvert au moyen de OPEN IN, OPEN OUT, APPEND, etc.
Fichier protégé contre l'écriture (90): Vous ne pouvez pas modifier le fichier sélectionné parce qu'il a été verrouillé intentionnellement par la commande PROTECT du CLI.
Fichier protégé contre l'effacement (89): Le disque système de l'Amiga vous permet de protéger chaque fichier contre l'effacement en utilisant une commande spéciale PROTECT du CLI. Vous avez sans doute essayé d'effacer un fichier système important avec DELETE.
Fin du programme (10): Ce message est imprimé à l'écran après qu'AMOS ait exécuté la dernière instruction de votre programme.
Format de fichier non reconnu (95): La commande LOAD peut seulement être utilisée pour entrer des banques mémoire AMOS depuis le disque. Il se peut que vous ayez confondu LOAD et LOAD IFF (charge un écran). Il vous faut utiliser l'instruction BLOAD si vous voulez charger un fichier de données sauvé dans le format Amiga.
Icône non défini (74): L'icône que vous spécifiez dans votre instruction est introuvable dans la banque courante d'icônes (banque 2).
Image trop grande pour l'écran courant (32): Vous avez essayé de charger une image dans un écran existant à l'aide de LOAD IFF, mais les deux écrans ne sont pas du même type. Affectez le numéro de l'écran de destination à la commande LOAD IFF comme suit:

```
Load Iff "nomfichier",numéro
```

A condition que l'écran que vous avez précisé dans le numéro se trouve dans la bonne plage (de 0 à 7), AMOS créera automatiquement un écran du type demandé lors du chargement.
Impossible de passer en DUAL PLAYFIELD (70): Vous avez essayé de créer un double champ de lecture en utilisant le mauvais type d'écran. Voir le chapitre sur la commande DUAL PLAYFIELD qui donne une liste de combinaisons autorisées.
Label non défini (40): L'étiquette comprise dans votre instruction n'a pas été définie dans votre programme. Recherchez les erreurs dans les instructions calculées GOTO, GOSUB ou RESTORE.
Les écrans ne sont pas ANIMables (67): AMAL peut seulement déplacer ou faire défiler des écrans. Il n'est pas possible de les animer au moyen d'une commande Anim.
Liste copper trop longue (77): Par défaut, la liste de Copper définie par l'utilisateur est limitée à un maximum de 12k. Elle peut être prolongée au moyen d'une option se trouvant dans l'accessoire CONFIG.
Mauvais format IFF (30): Vous avez essayé d'utiliser LOAD IFF pour charger un fichier qui a été rangé dans un format inhabituel. Rappelez-vous que LOAD IFF peut seulement charger les écrans dans la mémoire et non des fichiers IFF universels.

Mémoire pleine (24): C'est le message d'erreur standard qui se produit lorsque vous essayez de dépasser la capacité de la mémoire. Ne paniquez pas: Il existe trois façons de conserver la mémoire. Voici une liste:

1 CLOSE WORKBENCH désactive le système workbench de l'Amiga pour libérer 40k.
2 CLOSE EDITOR désactive la fenêtre de l'éditeur lorsque celle-ci n'est pas utilisée.Ceci économise 24k!
3 Si la ligne d'information signale qu'il y a beaucoup de mémoire libre, sauvez votre programme et réamorcez-le. Ceci corrigera un problème de fragmentation de la mémoire causé par le système d'exploitation de l'Amiga.

Menu non ouvert (38): La commande MENU ON a été appelée mais il n'y a pas de menu. Vous avez d'abord besoin de définir votre menu en utilisant l'instruction MENU$ ou MAKE MENU BANK.
Nom de fichier interdit (82): Vous avez essayé d'utiliser un nom de fichier qui n'est pas conforme aux règles de nomination standards. Référez-vous au manuel d'utilisation qui vous a été livré avec votre Amiga.
Numéros d'écrans valides entre 0 et 7 (50): AMOS vous permet seulement d'ouvrir un maximum de huit écrans à la fois.
Objet de menu non défini (39): L'article spécifié dans la commande de votre menu n'a pas été préalablement défini à l'aide de MENU$.
Paramètre copper interdit (78): La valeur que vous avez entrée dans une instruction COP MOVE, COP MOVEL ou COP SWAP n'est pas comprise dans la plage permise.
Paramètres de bloc interdits (66): Vous avez commis une erreur dans la commande GET BLOCK ou PUT BLOCK. Les valeurs que vous avez entrées ne sont pas permises.
Paramètre d'écran interdit (48): Les dimensions que vous avez spécifié à l'aide de SCREEN OPEN ne sont pas admissibles. La taille minimum d'écran est de 32 sur 8 et la taille maximale dépend entièrement de la quantité de mémoire "chip" disponible.
Paramètre de fenêtre interdit (60): Vous avez entré une valeur incorrecte dans une des commandes fenêtre.
Pas de DATAs apre`s cette etiquette (41): RESTORE a essayé de déplacer le pointeur de données vers une ligne ne comprenant pas d'instructions DATA.
Pas de disquette dans le lecteur (93): Vous avez essayé d'accéder au lecteur qui ne semble pas contenir une disquette. Si vous venez de placer une disquette dans le lecteur, attendez quelques secondes et essayez de nouveau.
Pas de ON ERROR PROC auparavant (5): RESUME LABEL est seulement permis après une commande ON ERROR PROC.
Pas de programme sous le programme courant (43): Si le programme qui est en train de se dérouler n'a pas été installé en tant qu'accessoire et si vous essayez de saisir une banque à l'aide de BGRAB, vous obtiendrez ce message.
Répertoire introuvable (80): Le répertoire demandé n'existe pas sur le disque courant. Il se peut que vous ayez inséré la mauvaise disquette par inadvertance.
Répertoire non vide (85): Il est seulement possible d'effacer des répertoires VIDES à l'aide de SUPPRIMER.
Périphérique indisponible (86): Le disque ou l'unité que vous avez spécifiée dans l'instruction n'est pas branchée sur l'Amiga. Ce message d'erreur est souvent causé par un changement de disque inattendu. Pour résoudre ce problème, affectez le nom du

lecteur existant au répertoire en utilisant une ligne comme suit:

```
Dir$="Df0:"
```

Si ce message se produit lors de l'utilisation du sélecteur de fichier, cliquez sur un des boutons de commande tels que "Df0:" et changez le répertoire en utilisant le bouton "Set Dir".

Plus de données (33): La commande READ a essayé de lire ce qui se trouve après le dernier article DATA de votre programme. Vous avez sans doute omis des informations lorsque vous étiez en train d'entrer une de vos lignes DATA. Recherchez également les fautes de frappe dans toutes les commandes RESTORE.

Plus d'espace de chaînes (11): Par défaut, AMOS alloue seulement 8k d'espace de rangement pour vos chaînes et tableaux. Utilisez une commande SET BUFFER au début de votre programme pour augmenter ce nombre si besoin.

Plus d'espace de la pile (0): Ce message d'erreur est produit lorsque vous essayez d'emboîter trop d'appels de procédure. Les procédures AMOS peuvent s'appeler (récurrence), mais vous obtiendrez un message d'erreur après la 50ème boucle.

Polices non examinées (37): Avant d'utiliser la commande SET FONT, vous devez d'abord créer une liste des polices à l'aide de GET FONTS, GET DISC FONTS ou GET ROM FONTS.Nombre de couleurs invalide (49): La machine de l'Amiga ne peut pas supporter en même temps toutes les combinaisons de couleurs à l'écran. Voir le chapitre sur ECRANS pour une liste complète des options disponibles. Vous avez sans doute fait une faute de frappe dans la commande SCREEN OPEN.

POP sans GOSUB (2): POP peut seulement s'exécuter dans une sous-routine qui a été préalablement lancée à l'aide de GOSUB. Utilisez POP PROC pour sortir d'une procédure.

Programme interrompu (9): Ce n'est pas une erreur; il vous suffit d'appuyer sur les touches Cntrl-C ou d'utiliser une instruction STOP pour sortir directement de votre programme.

Programme introuvable (42): Le programme nommé dans la commande PRUN n'a pas été préalablement entré dans la mémoire de l'Amiga. Utilisez LOAD OTHER de la fenêtre MENU.

RAINBOW non défini (75): Avant d'appeler la commande RAINBOW dans un de vos programmes, vous devez d'abord définir votre arc-en-ciel en utilisant SET RAINBOW.

RESUME interdit sur un label (4): Vous ne pouvez pas utiliser l'étiquette RESUME dans une procédure d'erreur.

RESUME label non définie (6): L'étiquette que vous avez spécifiée dans la commande RESUME n'existe pas.

RESUME sans erreur (7): La commande RESUME ne peut pas s'exécuter si une erreur ne s'est pas produite dans votre programme. Il est préférable de l'utiliser pour revenir sur l'AMOS Basic après qu'une erreur ait été détectée.

RETURN sans GOSUB (1): Vous pouvez seulement utiliser RETURN pour sortir d'une sous-routine qui a été d'abord lancée à l'aide de GOSUB.

Tableau déjà dimensionné (28): Vous avez essayé de dimensionner un tableau deux fois dans votre programme. Cette erreur est d'habitude détectée lors de la vérificaton initiale de la syntaxe, mais si votre programme est compliqué, elle sera seulement découverte lorsque vous essayez de redimensionner votre tableau.

Tableau non dimensionné (27): Votre programme a essayé de se reporter à un tableau qui n'a pas été préalablement défini.
Trop de couleurs dans le Flash (51): Vous avez dépassé le nombre maximal de 16 changements de couleurs dans une seule commande FLASH.
Type de variables incompatibles (34): Une valeur interdite a été affectée à une variable. Par exemple:

```
A$=12 devrait se lire A$="12"
```

Type de compression IFF inconnue (31): L'écran que vous souhaitez charger depuis le disque emploie un système de compactage qu'il ne connaît pas. Renvoyez-nous le progiciel graphique que vous utilisez pour créer cet écran et sauvegardez-le dans le format IFF.
Type de fichier incompatible (98): Une commande du disque a été utilisée, ce qui n'est pas autorisé dans le fichier courant. L'erreur se produit si vous devez utiliser les commandes GET et PUT pour accéder au fichier séquentiel.
Une procedure d'erreur doit se finir par un RESUME (8): Vous ne pouvez pas sortir d'une procédure de gestion de l'erreur à l'aide de END PROC. Vous devez utiliser une des commandes spéciales RESUME.
Zone de scrolling non définie (72): Avant d'utiliser la commande SCROLL, vous devez définir la direction et la taille de votre zone de défilement à l'aide de SET SCROLL.

Erreurs dans les fichiers d'extension

Les commandes qui entrent dans les fichiers d'extension AMOS ne produisent pas de numéros d'erreur mais vous devez connaître leur signification. Voici les erreurs des fichiers d'extension MUSIC:

256 caractères par onde: Vous pouvez seulement créer des ondes avec un bloc de 256 octets.
Banque d'échantillons introuvable: Il n'y a pas de banque d'échantillons en mémoire.
Banque musique introuvable: La musique ne peut pas être jouée parce qu'il n'y pas de mélodie en mémoire.
Banque musique indéfinie: Il n'y a pas de banque musique en mémoire.
Impossibilité d'ouvrir le synthétiseur de parole: AMOS ne peut pas trouver les fichiers demandés de la bibliothèque du disque système pour charger le programme synthétiseur.
Les ondes 0 et 1 sont réservées: Ces deux ondes sont réservées par AMOS pour les commandes BELL et NOISE et vous ne pouvez donc pas les changer.

L'extension de compactage peut créer les deux erreurs suivantes:

Ce n'est pas un écran comprimé: Les données que vous essayez de décondenser ne se trouvent pas dans un format écran comprimé.
Ce n'est pas une bitmap compactée: Vous avez essayé de décondenser une banque de données qui n'est pas dans un format de bitmap.

Command Index

Indexe de commandes

MANDARIN
SOFTWARE