ToolBox Mag

n° 3

Février - Mars 1991

Sommaire de la revue :

Prix: 590F par abonnement.

Présentation de la disquette ToolBox-Mag No 3 (🍏)

Nous ne nous lasserons jamais de le rappeler, ToolBox Mag se lit GS en route: **lisez la disquette!** Voici son catalogue:

/ANIM.03/: ce catalogue contient les fichiers nécessaires à la présentation de ToolBox Mag, le magazine qui boote sur GS. N'y touchez pas.
/A.LIRE/: les articles sur la disquette, en format Ascii étendu et en format AppleWorks-GS. A vous de les imprimer si besoin.
• **CONVSEC:** un article de notre agent X27 sur une nouvelle fonction du Miscellaneous Toolset.
• **MAC.GS:** un article de JYB à propos d'une polémique sur le LC.
• **VOC2:** un encadré de JYB à propos de la Video Overlay Card.
• **WANTED:** un courrier à deux voix (A. Bonnet et JYB) de suggestions aux programmeurs.
/C.FACILE/: contient les ossements du "squelette C" de François Uhrich. Voir les pages 4 à 8, et le fichier Lisez.Moi de ce catalogue.
/DERNIERE.HEURE/: si Apple abandonne le Macintosh avant le 20 Janvier 91, vous le saurez en lisant le Lisez.Moi de ce catalogue.
/GOODIES/: ce dossier contient des programmes dont l'utilité nous semble un peu marginale ou exotique, mais qui tournent sur GS: nous n'avions pas de raison de refuser de les publier, et ils seront utiles à quelques-uns d'entre vous. (Attention, à la différence de ce qui se passe dans d'autres magazines, les "Goodies ToolBox Mag" sont des programmes ToolBox Mag, couverts comme les autres par notre inflexible Copyright). A chaque fois, un fichier Lisez.Moi fournit les explications nécessaires.
• **CDA.LOAD:** init d'Yvan Kœnig permettant de charger des CDA très, très, tôt.
• **MOUSETEXT.AWP:** un programme, écrit par Didier Skutnik, de patches à Appleworks-8 version 3.0 française, et un fichier de macros, pour y introduire les caractères Mousetext.
• **OTL:** trois "2-Liners" en Applesoft pur jus par Pierre Raynaud-Richard. L'Arkanoïd 2 en deux lignes nous a bien amusés.
/ICONS/: icônes pour le Finder.
/JAQUETTES.K7/ et **/JYB.FONTES/**: voyez page 47 (J.Y. Bourdin).
/MAC.ERASER/: voyez page 28 (O. Goguel).
/MASTERMIND/: le jeu, et son source, de Raoul Barbieux. Voyez pages 29 et suivantes.
/PLATINUM.PAINT/: illustration pour l'article d'Eric Weyland pages 48/49.
/RESSOURCES/: le squelette de notre "ressourcier" Bernard Fournier. Voir pages 37 et suivantes.
/TOOL219.V.1.1/: la nouvelle version du SoundTrack Tool d'Olivier Goguel, et presque tous les interfaces imaginables pour cet outil, par A. Cassaigne et Y. Kœnig. Voir page 42.
/TRACEUR.NDA/: l'accessoire-espion de Patrick Desnoues, avec ses ressources et librairies. Lisez (lentement et à tête reposée) l'encadré page 15.
/VIDEO.LIAUTARD/: illustrations de l'article sur le GS et la vidéo de J. Liautard pages 21 à 26.

Finder.Root est un fichier pour le Finder, **PRODOS** est un faux.

ToolBox-Mag

Directeur de la Publication
Eric Weyland

Rédacteur en Chef
Jean-Yves Bourdin

Mise en pages
Dominique Digoy

Secrétariat
Janine Loiseleux

Conseil de Rédaction
Jean-Pierre Charpentier
Bernard Fournier
Olivier Goguel
Stéphan Hadinger
François Hermellin
Yvan Kœnig
Hubert Loiseleux
Claude Pélisson
Jean-Luc Schmitt
François Uhrich

Rédaction, Siège Social
ToolBox-Mag
6, rue Henri-Barbusse
95100 Argenteuil - France
Téléphone: (1) 30 76 18 64
Télécopie: (1) 39 47 44 08

Impression
Imprimerie /Reprographie /Graphisme - Argenteuil

Les auteurs de ToolBox-Mag n° 3

Olivier Bailly-Maître
Raoul Barbieux
Jean-Yves Bourdin
Anthony Cassaigne
Patrick Desnoues
Bernard Fournier
Olivier Goguel
Yvan Kœnig
Jacques Liautard
Serge Mousseron
Pierre Raynaud-Richard
Didier Skutnik
François Uhrich
Eric Weyland

Avertissements légaux

ToolBox-Mag est un magazine qui prend l'Apple II GS au sérieux. De ce fait même, il est indépendant d'Apple Computer et d'Apple Computer France, auxquels il n'est, grâce au ciel, pas rattaché.

ToolBox-Mag est disponible uniquement par abonnement et par correspondance. Le magazine est composé inséparablement d'une revue sur papier et d'une disquette magnétique pour ordinateur Apple // GS. Certains avertissements légaux, modes d'emploi, etc, ou tous autres textes, peuvent être placés sur la disquette, et nécessiter un Apple II GS pour être lus: ils n'en sont pas moins écrits dans ToolBox-Mag, dont ils font partie intégrante.

Tout le contenu de la disquette et de la revue ToolBox-Mag sont entièrement sous Copyright de ToolBox-Mag, et sont déposés à l'Agence pour la Protection des Programmes. Tous droits de traduction et de reproduction intégralement réservés pour tous pays.

Aucune reproduction, même partielle, et sous quelque forme que ce soit, de la disquette comme de la revue ToolBox-Mag, ou d'un des programmes qu'elles contiennent, ne pourra avoir lieu sans accord préalable et écrit de ToolBox-Mag. Copier ToolBox-Mag, c'est illégal, et cela vise à torpiller le seul magazine français sur l'Apple // GS; nous serons alors sans pitié.

"GS" est un sigle déposé par les automobiles Citroën. La Pomme est un symbole déposé par Eve et le Serpent. Apple, Apple II GS, le logo Apple, Macintosh et le logo Macintosh, ImageWriter, LaserWriter et 'Fatal System Error 911' sont des marques déposées d'Apple Computer. AppleWorks-GS est une marque déposée par Claris-USA, à l'insu de Claris France. ToolBox™ et ToolBox-Mag™ sont des marques déposées de la Société ToolBox®.

ToolBox-Mag ne peut pas offrir une garantie supérieure à celle qu'offre le chapitre "Limitations de Garantie et de Responsabilité" des documentations officielles Apple, soit quasiment rien. Ecrit par des amateurs du GS, il s'engage néanmoins à publier des programmes ne contenant pas plus de bugs que GS Write, et à maintenir dans ses colonnes un standard de compétence au moins égal au niveau moyen des concessionnaires Apple français en matière de GS...

EDITORIAL (ℭ)

La passion et le profit

On n'y croyait plus: un magazine dédié aux amoureux de l'Apple II GS qui n'est pas le 'fanzine' d'un club, mais bel et bien un **produit commercial d'une entreprise commerciale**, c'est possible?

En faire la preuve, telle est notre ambition. Les passionnés du GS ne sont pas des boy-scouts naïfs. Ils savent qu'une machine, aussi géniale soit-elle par elle-même, ne peut vivre que si ses utilisateurs en assurent le succès: et c'est en termes de bilan comptable, en termes de 'gros sous' que se mesure ce succès.

Les réalités de l'industrie étant ce qu'elles sont, ToolBox Mag est un sacré "challenge": il s'agit d'**assurer le succès commercial d'un magazine dédié aux utilisateurs d'un ordinateur, sans aucune rentrée publicitaire, et sans aucun soutien du constructeur.** Défi plutôt inédit dans la presse informatique, vous en conviendrez!

Utilisateurs de GS, vous n'êtes pas qu'une bande de joyeux loustics: **vous êtes un marché.** Le pari de ToolBox Mag *(et de son éditeur, l'entreprise qui a accepté de perdre de l'argent à coup sûr tout de suite pour en gagner peut-être ensuite, ToolBox)*, c'est de prouver que **ce marché est rentable**, et que si tant d'autres l'ont abandonné, c'est faute de faire les produits de qualité qu'attendent ses passionnés.

Voilà pourquoi notre magazine porte, dans son nom même, la marque commerciale de l'entreprise ToolBox®, comme l'Apple II GS porte le nom de son constructeur, ou le Petit Larousse celui de son éditeur. **C'est le nom de ceux qui ont osé le faire, qui ont osé prendre le GS au sérieux au plan commercial**, comme ToolBox Mag le prend au sérieux en tant qu'ordinateur personnel.

Cela dit, si le profit était le premier souci de ToolBox Mag, ce serait un magazine "comme les autres", pour IBM et compatibles, ou pour Macintosh. On ne risque guère, en France aujourd'hui, de faire fortune avec le GS! C'est donc une **autre raison** qui anime notre équipe de rédaction et de publication: la passion, **le plaisir, bien sûr! Le profit, c'est simplement la condition pour continuer le plaisir.**

Pas étonnant donc que nous défendions nos droits et copyrights: s'il y a des magazines qui sont de purs supports publicitaires, qu'on nous payera peut-être un jour pour lire, ToolBox Mag n'est pas du nombre. Ceux qui le copient travaillent à couler le seul magazine français pour le GS, c'est aussi simple que cela. **Nous ne les laisserons pas faire: nous aimons trop nos GS!**

Certes, la rentabilité, nous n'y sommes pas encore, vous le savez, et ce "satané Directeur" doit continuer à compter sou par sou. Mais nous nous en rapprochons, plus vite que nous ne le pensions. En dernier ressort, ce seront **nos clients**, les utilisateurs de GS, qui décideront de la réussite ou de l'échec commercial de notre pari, comme d'ailleurs de l'avenir de leur machine. **Nous avons confiance: vous aimez trop vos GS!**

ERIC WEYLAND

C facile n°3 : un squelette dans la Boîte à Outils ! ( )

François Uhrich

Aujourd'hui, nous allons entrer dans la Boîte à Outils avec le langage C.

Tout cet article va décrire le *squelette* d'un programme C de type desktop. Pourquoi le terme *squelette*? Tout simplement parce qu'à partir de ce programme minimal, vous pourrez écrire tous les vôtres en vous basant sur la même *structure* de départ.

Comme d'habitude vous trouverez sur la disquette le source complet de ce programme, écrit de telle façon qu'il puisse être compilé indifféremment avec Orca/C™ ou APWC. Vous constaterez d'ailleurs le peu de différence entre les deux versions (lignes avec __ORCAC__).

Ce programme affiche uniquement une barre de menus comportant le menu Pomme, le menu Fichier et le menu Edition. Son seul intérêt *pratique* est qu'il permet l'utilisation des Accessoires de Bureau; son intérêt *pédagogique* est je crois plus élevé...

Grâce à l'utilisation des ressources, le programme C est très bref. Commençons sa description détaillée.

Les Includes

Comme tous les programmes en C, quelle que soit la machine, notre squelette commence par des includes:

```
#include  <types.h>
#include  <locator.h>
#include  <memory.h>
#include  <misctool.h>
#include  <quickdraw.h>
#include  <menu.h>
#include  <desk.h>
#include  <resources.h>
```

La commande préprocesseur *#include <entete.h>* inclut le source C *entete.h* (h pour header) situé en **2/OrcaCDefs** pour Orca/C et **2/CInclude** pour APWC. Pour inclure un fichier au chemin d'accès quelconque au lieu de <>, il faut utiliser: *#include "MonChemin"*.

Tous ces fichiers **.h** contiennent soit la définition des structures et des fonctions contenues dans la librairie C (comme par exemple stdio.h), et plus ou moins communes à tous les compilateurs C sur toutes les machines, soit la définition des structures de données et des fonctions de chacun des Outils de la Toolbox (comme par exemple window.h), donc spécifiques au GS.

La difficulté réside dans le choix des en-têtes à inclure: d'une part, s'il en manque, un message d'erreur vous sera retourné par le compilateur (structure non définie) ou le linker (fonction non définie); d'autre part, s'il y en a trop, alors le temps de compilation peut devenir bien long...

La bonne méthode consiste à ajouter un en-tête dès qu'on utilise une structure ou une fonction faisant partie d'un Outil: j'écris dans mon programme *MMStartUp()*, alors au début du programme j'ajoute *#include <memory.h>*. Cette méthode demande bien sûr une bonne connaissance de la Toolbox, mais avec les 3 volumes de référence de la Toolbox sous la main, on peut très bien se débrouiller.

Je vous recommande de regarder ces en-têtes, car vous constaterez qu'ils sont écrits de façon à pouvoir s'inclure les uns les autres. Ainsi, au début de *Desk.h*, on trouve:

```
#ifndef __TYPES__
#include <TYPES.h>
#endif

#ifndef __QUICKDRAW__
#include <QUICKDRAW.h>
#endif

#ifndef __EVENT__
#include <EVENT.h>
#endif

#ifndef __DESK__
```

NDLR: C du C, ça ?

Vous qui n'aimez pas l'octal, vous chez qui le mot de "C" évoque galère, Unix, MS/Dos & C°, bref vous qui sautez allègrement d'habitude la rubrique de François Uhrich: ***S'il vous plaît, lisez celle-ci!***
Un squelette d'application desktop en C ressemble comme deux gouttes d'eau à un squelette en autre chose: c'est de la ToolBox GS avant tout.
Ce C-là, ce "ToolBox-C", n'est pas portable ailleurs, même pas sur Mac. C'est du GS 100%. C'est bon pour tout le monde. Allez, courage!

JYB

#define __DESK__

Ainsi la simple ligne *#include <desk.h>* entraînera l'inclusion de *Types.h, QuickDraw.h, Event.h* si ces derniers n'ont pas déjà été inclus. On peut donc optimiser au maximum la liste des includes.

#Define

Après les *#includes*, voici les *#define*:

```
#define Edit_Menu  0x00000003
#define Close_Item 0x000000FF
#define Quit_Item  0x00000100
#define About_Item 0x0000012D
```

#define est une commande préprocesseur (comme toutes celles commençant par #) très puissante, car elle permet la définition de constantes en leur affectant une chaîne de caractères pouvant aussi bien comprendre des nombres que du texte. Avant la compilation proprement dite (d'où le terme de commande préprocesseur), le compilateur remplacera dans le texte toutes les occurrences des définitions par leur chaîne correspondante. Ainsi dans le texte de notre programme, chaque *Edit_Menu* sera remplacé par *0x00000003*, c'est à dire pour le compilateur la valeur 3.

Il s'agit d'un remplacement **intelligent**, car ces définitions peuvent devenir de véritables **macro instructions.** Si, par exemple, on a posé:
#define sum(a,b) (a+b),
et si dans le programme, on a : *x = sum(y,z);*
le préprocesseur remplacera cette ligne par: *x = (y+z);*
Je vous laisse imaginer la puissance que peuvent procurer ces macro instructions...

Dans notre programme, 3 est le numéro du menu d'Edition, 255 (en hexadécimal 0xFF) le numéro de l'article Fermer, 256 (0x100) le numéro de l'article Quitter et 301 (0x12D) le numéro de l'article "A propos" tels qu'ils sont numérotés dans les ressources de

Les fonctions C

La déclaration d'une fonction commence par le type du résultat. Si ce type est *void*, alors c'est qu'il n'y a pas de résultat, ce qui correspond donc à une procédure pascalienne: *void MaProcedure()*
Même si un résultat est prévu dans la déclaration, celui-ci peut être ignoré, ce qui revient au type *void*.
Si le type n'est pas indiqué, alors par défaut le résultat sera un entier *int*: *int MaFonction()* est équivalent à *MaFonction()*.

Suit le nom de la fonction et, entre parenthèses, ses arguments sont énumérés, avec, dans les lignes suivantes, le type de chacun des arguments, avant de commencer enfin le bloc d'instructions de la fonction:

```
int MaFonction(arg1, arg2, arg3)
int arg1, arg3;
long arg2;
{
/* bloc d'instructions de MaFonction */
  return(result);
}
```

Pour retourner le résultat, l'instruction *return* doit être utilisée.

Dans le source du programme étudié, vous pouvez constater ce qui peut paraître au départ une anomalie dans le passage des paramètres de la fonction ExecMenu. En effet, cette fonction est appelée par l'instruction *ExecMenu(tache.wmTaskData)* alors que sa déclaration indique ***deux*** arguments:

```
ExecMenu(art,menu)
int art,menu;
```

Un compilateur Pascal rencontrant une telle anomalie, déclencherait toutes ses sirènes d'alerte...Tout s'explique par le fait qu'en C ce n'est pas le nombre de paramètres qui importe, mais la *taille en octets* des paramètres donnés. Ainsi le champ *wmTaskData* de la structure *tache* est un *long* (quatre octets) et les deux arguments attendus sont deux *entiers* (chacun composé de deux octets) ! L'argument *art* aura donc pour valeur les deux premiers entiers de *wmTaskData* et l'argument *menu* les deux derniers.

Voilà un exemple qui illustre une des plus importantes différences entre le C et le Pascal: le C est beaucoup moins rigoureux que le Pascal, mais c'est au programmeur de l'être un peu plus ! A mon avis, cela permet avec le C beaucoup plus de liberté, donc de puissance, qu'avec le Pascal.

NDLR: s'il vous plaît, lecteurs, évitez le déluge de courrier à propos de cette dernière phrase...

ces menus.

Les Variables Globales

Les variables globales ont l'avantage d'être utilisables dans toutes les fonctions du programme, leur inconvénient majeur est qu'elles occupent donc de la mémoire durant toute l'exécution du programme. Pour celui qui nous concerne, il y en a très peu:

```
Boolean   fin;
word      myID;
```

le booléen *fin* aura la valeur VRAI lorsque l'utilisateur aura sélectionné l'article Quitter du menu Fichier et l'entier *myID* contiendra l'identificateur mémoire du programme tel qu'il est attribué par le Memory Manager. Les demandes d'allocations mémoire au Memory Manager devront toujours se faire avec cette valeur, voilà pourquoi elle est définie comme globale même si dans notre exemple elle n'est utilisée que dans la fonction *main()*.

Il y a enfin la chaîne de caractères *compiler* qui sera utilisée dans le "A propos" pour indiquer le compilateur qui a été utilisé: soit "Orca/C™", soit "APWC".

```
#ifdef __ORCAC__
char  compiler[] = "ORCA/C\252";
#else
char  compiler[] = "APWC";
#endif
```

Je rappelle que *__ORCAC__* vaut 1 si le compilateur est Orca/C™ et 0 sinon.

Ouverture de la Boîte à Outils

Les variables locales (*code, ToolRecRef, oldwin, tache*) étant déclarées, le programme commence par l'ouverture des Outils.

Le système doit en effet être averti qu'un programme veut utiliser tel ou tel Outil. Cet Outil doit être ouvert en début de programme et fermé à la fin. De plus un ordre strict d'ouverture et de fermeture (inverse du premier) doit être scrupuleusement respecté. Enfin certains de ces Outils nécessitent la réservation d'un espace mémoire de taille précise en banc mémoire 0.

Il fut un temps où toutes ces opérations nécessitaient plusieurs pages de programme. Heureusement, depuis le système 5.0 sont apparues les fonctions *StartUpTools* et *ShutDownTools* qui permettent de réduire considérablement la consommation d'aspirine chez le programmeur !

```
myID = MMStartUp();
ToolRecRef=StartUpTools(myID,refIsResource,1L);
```

Tout d'abord le Memory Manager est ouvert par *MMStartUp* qui retourne dans *myID* l'identificateur mémoire de notre programme.

Ensuite la fonction *StartUpTools* est appelée avec comme premier paramètre *myID*, qui sera utilisé pour l'allocation mémoire en banc 0. Le deuxième paramètre a pour seule utilité d'indiquer le type du troisième: celui-ci sera donc dans ce cas une ressource.
De nombreuses nouvelles fonctions de la Toolbox (décrites en détail dans le volume 3 du Manuel de Référence) utilisent ce système de paramètre indicateur. Il y a couramment trois valeurs possibles: 0 ou *refIsPointer*, 1 ou *refIsHandle* et enfin 2 ou *refIsResource*.
Voici la ressource en question en langage REZ:

```
resource rToolStartup (1, $0050) {
    $C080, {          // videoMode
        3, $0300,     // Miscellaneous Tools
        4, $0301,     // QuickDraw II
        5, $0302,     // Desk Manager
        6, $0300,     // Event Manager
        11, $0200,    // Integer Math
        14, $0301,    // Window Manager
        15, $0301,    // Menu Manager
        16, $0301,    // Control Manager
        18, $0301,    // QuickDraw II Aux.
        20, $0301,    // LineEdit Tools
        21, $0301,    // Dialog Manager
        22, $0300,    // Scrap Manager
        27, $0301,    // Font Manager
        30, $0100     // Resource Manager
    }
};
```

A côté du numéro d'outil, l'entier en hexadécimal indique le numéro de version minimal de cet Outil acceptable pour pouvoir exécuter ce programme: 5, $0302 indique que le Desk Manager doit être au moins la version 3.02.

Le résultat de *StartUpTools* est un handle dans le cas où le troisième paramètre est une ressource ou un handle, sinon il s'agit d'un pointeur. Ce résultat est à conserver, car il sera utilisé au moment du *ShutDownTools:*

```
ShutDownTools(refIsHandle,ToolRecRef);
MMShutDown(myID);
```

Création de la Barre de Menus

```
SetSysBar(NewMenuBar2(refIsResource,1L,0L));
SetMenuBar(0L);
FixAppleMenu(1);
FixMenuBar();
DrawMenuBar();
```

NewMenuBar2 crée une barre de menus à partir, dans notre cas, de la ressource suivante:

```
resource rMenuBar (1, $0000) {
    {
    Apple_Menu,
    File_Menu,
    Edit_Menu
    };
};
```

Voir le source en langage REZ, et l'article de B. Fournier dans ce même numéro de ToolBox Mag, pour détailler toutes les ressources rMenu et rMenuItem utilisées dans cette barre de menus. Le menu n°1 ou Apple_Menu est ainsi défini:

```
resource rMenu (Apple_Menu, $0000) {
    $0001,              // menuID
    $A008,              // menuFlag
    Apple_String, {     // menuTitleRef
        About_Item
    };
};
```

Avec :

```
resource rMenuItem (About_Item, $0000) {
    $012D,              // itemID
    "","",              // itemChar, itemAltChar
    NIL,                // itemCheck
    $8040,              // itemFlag
    About_String        // itemTitleRef
};
```

SetSysBar permet de faire de cette nouvelle barre de menus, la barre système; *SetMenuBar(0L)* indique que c'est la barre système qui est active; *FixAppleMenu(1)* ajoute à la barre active au menu n°1 (ici le menu Pomme) le menu des NDA; *FixMenuBar* effectue les calculs sur les dimensions des menus; enfin (!) *DrawMenuBar* dessine cette barre à l'écran.

Après la création de la barre de menus, certaines variables sont initialisées et le curseur **flèche** est affiché (car *StartUpTools* affiche le curseur **montre**):

```
tache.wmTaskMask = 0x001FFFFF;
FlushEvents(everyEvent,0);
oldwin = (GrafPortPtr)-1L;
fin = FALSE;
InitCursor();
```

La boucle des événements

Nous entrons alors dans une boucle exécutée continuellement tant que le booléen *fin* n'est pas VRAI:

```
do {

} while (!fin);
```

Il s'agit là du **cœur de tous les programmes desktop: la boucle des événements.**

La boucle commence ici par la gestion des articles des menus actifs ou non. Dans notre cas plutôt réduit, les seuls articles susceptibles de changer d'état sont ceux du menu Edition (Annuler, Couper, Copier, Coller et Effacer) ainsi que l'article Fermer du menu Fichier. Tous ces articles ne seront actifs que dans un seul cas: si une fenêtre d'un NDA est au premier plan sur le desktop:

```
if (oldwin != FrontWindow()) {
    oldwin = FrontWindow();
    if ((oldwin!=NIL)&&GetSysWFlag(oldwin)) {
        SetMenuFlag(enableMenu,Edit_Menu);
        EnableMItem(Close_Item);
    }
    else {
        SetMenuFlag(disableMenu,Edit_Menu);
        DisableMItem(Close_Item);
    }
    HiliteMenu(FALSE,Edit_Menu);
}
```

La variable *oldwin* (initialisée à -1) contient le pointeur de la dernière fenêtre au premier plan, et la fonction *FrontWindow* retourne le pointeur de la fenêtre actuellement au premier plan; si ces deux valeurs ne sont pas les mêmes, alors une nouvelle fenêtre a été ouverte (si la *FrontWindow* n'est pas nulle, car sinon cela signifie qu'il n'y a pas de fenêtre sur le desktop).

Si une fenêtre est présente et qu'elle est du type système (appel à *GetSysWFlag*) alors il s'agit d'une fenêtre d'un NDA: le menu *Edition* est donc rendu actif (*SetMenuFlag*) ainsi que l'article *Fermer* (*EnableMItem*). Dans les autres cas, ils sont rendus inactifs (*SetMenuFlag* et *DisableMItem*).

Enfin ces modifications sont prises en compte à l'affichage par un *HiliteMenu* sur le menu *Edition* (de préférence à un *DrawMenuBar* inconfortable pour l'œil de l'utilisateur, voir TechNote IIGS n°60).

Nous arrivons enfin à la fonction majeure de la Boîte à Outils de l'Apple IIGS, celle qui fait tout pour nous, le magnifique TaskMaster.

La merveille des merveilles: TaskMaster

Expliquer en détail tout ce que fait *TaskMaster* occuperait pratiquement toutes les pages de ce Toolbox Mag. Le Rédacteur en Chef étant un peu réticent (!), je vais tenter de résumer ce que fait la Bête: *TaskMaster* fait **TOUT sauf** ce qui doit être fait par l'application courante ! (Merci La Palice...)

```
code = TaskMaster(everyEvent,&tache);
```

Le premier paramètre est un entier indiquant à la fonction quels sont les événements dont elle doit s'occuper; dans notre cas *0xFFFF* ou *everyEvent*, donc de tous les événements: souris enfoncée ou relâchée, touche appuyée ou répétition, mise à jour d'une fenêtre... etc.

Le deuxième paramètre est l'adresse (c'est à dire le pointeur) de la structure manipulée par *TaskMaster* qui contiendra des informations capitales pour l'application: fenêtre concernée par l'événement, contrôle dans lequel l'utilisateur a cliqué, touche appuyée, nombre de clic(s) pour savoir s'il y a eu un double ou même triple clic; au moment de l'événement les touches Control, Majuscule, Option et Commande étaient-elles enfoncées, quelle était la position de la souris... **etc.**

Ainsi c'est *TaskMaster* qui fera tout le travail lorsque l'utilisateur sélectionnera un NDA, activera, déplace-

ra, fermera ou zoomera une fenêtre et bien d'autres choses ! ***Il n'y a aucune ligne de programme à écrire, cela fait partie intégrante de la Boîte à Outils !*** Il est à remarquer que cette fabuleuse fonction ne fait pas partie de la Boîte à Outils du Macintosh...

TaskMaster retourne, dans l'entier code, le type d'actions ou d'événements que l'application peut traiter: ainsi lorsque l'utilisateur sélectionnera un article dans un menu, *TaskMaster* retournera soit le code 25 (0x19) ou *wInSpecial* s'il s'agit d'un des articles particuliers du menu Edition, soit le code 17 (0x11) ou *wInMenuBar* dans tous les autres cas. Pour savoir quel est l'article sélectionné par l'utilisateur, il suffit d'aller regarder dans la structure tâche de type *WmTaskRec* ou *EventRecord* ainsi définie dans Event.h:

```
struct EventRecord {
  Word what; /* event code */
  LongWord message; /* event message */
  LongWord when; /* ticks since startup */
  Point where; /* mouse location */
  Word modifiers; /* modifier flags */
  LongWord wmTaskData; /*  */
  LongWord wmTaskMask; /*  */
  LongWord wmLastClickTick; /*  */
  Word wmClickCount; /*  */
  LongWord wmTaskData2; /*  */
  LongWord wmTaskData3; /*  */
  LongWord wmTaskData4; /*  */
  Point wmLastClickPt; /*  */
} ;
```

Ainsi dans le cas d'un *wInMenuBar* ou *wInSpecial*, on peut trouver dans le premier mot du champ *wmTaskData* le numéro de l'article sélectionné, et dans le deuxième mot de ce même champ le numéro du menu dans lequel a été sélectionné l'article.

```
switch(code) {
  case wInMenuBar :
  case wInSpecial :  ExecMenu(tache.wmTaskData);
                     break;
}
```

Il suffit donc au programme de réagir suivant l'article sélectionné. Dans notre cas, c'est la fonction *ExecMenu* qui réalise cela:

```
ExecMenu(art,menu)
int art,menu;
{
  char  *sub;
  switch(art) {
    case  Quit_Item  :  fin=TRUE;
                        break;
    case  About_Item  :  sub = compiler;
                         AlertWindow(4,&sub,1L);
                         break;
  }
  if (art) HiliteMenu(FALSE,menu);
}
```

Lorsque l'utilisateur sélectionnera l'article Quitter, le booléen *fin* deviendra VRAI, et donc la boucle des événements s'achèvera...

Les Outils sont alors fermés par *ShutDownTools* puis *MMShutDown*; la fonction principale *main* se termine et le programme s'achève par un appel *Quit* de GS/OS qui a été ajouté au moment de l'édition de liens par la librairie d'Orca/C™.

J'espère que ce survol rapide de la programmation événementielle en C vous a été agréable et que nous vous retrouverons dans deux mois sur nos lignes C !

Switch

L'instruction *switch* évalue l'expression entière donnée en argument et compare la valeur obtenue avec celle des différents cas (*case*) proposés. Si aucun de ceux-ci ne correspond à la valeur alors, s'il est présent, c'est le cas *default* qui sera pris en compte. Les cas doivent être soit des constantes entières ou de type caractère, soit une expression entière.
Une fois qu'un cas a été sélectionné et que la séquence d'instructions correspondante a été exécutée, le programme exécute alors toutes les séquences suivantes jusqu'à ce qu'il rencontre l'instruction *break*.
Cette particularité implique que l'instruction *switch* n'est pas équivalente à une succession de if-then-else sauf dans le cas où chaque séquence d'instructions se termine par un break.

```
switch(c) {
  case '0' :
  case '1' :
  case '2' :
  case '3' :
  case '4' :
  case '5' :
  case '6' :
  case '7' :
  case '8' :
  case '9' :
      printf("Chiffre : %c\n",c);
      break;

  default :
      printf("Erreur: ce n'est pas un chiffre\n");
      break;
}
```

Si le premier *break* n'était pas présent, les deux *printf* seraient exécutés quand le caractère c serait un chiffre... Le deuxième *break* n'est pas indispensable puisqu'il n'y a pas d'autre séquence, mais c'est une bonne habitude à prendre: si dans une future modification du programme de nouveaux cas sont ajoutés à la fin, ce *break* évitera de mauvaises surprises...

un accessoire espion : Traceur

Patrick Desnoues (ÓÓÓ)

Voici un nouvel accessoire de bureau, que j'ai hésité à diffuser, car il prend pour principe la dérivation des vecteurs de GS/OS, ce qui pourrait éventuellement donner des idées néfastes à des bidouilleurs malintentionnés. Mais il peut aussi donner de bonnes idées aux autres, en permettant d'améliorer GS/OS.

Il en va donc avec ce NDA Traceur comme avec toute connaissance: elle donne des armes pour le meilleur comme pour le pire. Mais justement, Toolbox-Mag a choisi de diffuser cette connaissance. Et comme Apple lui-même vient de publier une note technique concernant la revectorisation des outils (mais pas de GS/OS), on ne peut que se sentir incité à le faire.

Précisément, la valeur pédagogique de cet accessoire est essentielle. En affichant en clair et en temps réel toutes les commandes du système d'exploitation GS/OS, il permet à tout instant à l'utilisateur de savoir ce qui se passe dans sa machine.

A noter: cet accessoire contenant des ressources, il ne peut fonctionner qu'avec une version du système égale au moins à la 5.0. Le fait que Traceur modifie les vecteurs de GS/OS peut évidemment entraîner des incompatibilités avec des programmes qui utiliseraient le même principe...

Mode d'emploi du Traceur (Ó)

La mise en œuvre est très simple. Après avoir copié l'accessoire Traceur (qui se trouve en / ToolBox.Mag.03/ Traceur.NDA/ NDA) dans le dossier */system/ desk.accs de votre disque système, relancez votre GS par la commande adéquate du Finder. Notez que Traceur est compatible avec le Fonte/DA Installateur du Maître Uhrich, et peut en ce cas résider n'importe où, mais sur un disque qui restera en ligne (il a besoin de ses ressources).

Après cette installation, il suffit de cliquer sur la Pomme des menus du Finder pour appeler le Traceur.

Deux options sont alors disponibles:

☞ *Mettre une pause: Non*. Tous les appels au système seront alors affichés en temps réel. Particulièrement impressionnant sous le Finder ou le Standard File du 5.0 (tout spécialement celui du 5.03).

☞ *Mettre une pause: Oui*. Identique mais le système semblera tourner au ralenti, le rafraîchissement des fenêtres (sous Finder par exemple) ne s'effectuant qu'en appuyant un certain nombre de fois sur une touche du clavier. Une petite pomme à la droite de la ligne affichée signifie l'attente d'une touche.

• Pour activer le Traceur, il suffit de cliquer sur l'icône du milieu. Une fois le traceur activé, fermer la deuxième fenêtre active pour tout stopper.

• Pour le désactiver, fermer simplement l'accessoire.

☞ Avec certains programmes qui écrivent directement sur la page écran, la fenêtre du traceur peut disparaître. Il suffit d'appuyer sur la touche Option pour la redessiner.

Pourquoi cet accessoire? Eh bien, j'étais très curieux de voir la méthode utilisée par le Finder pour reconnaître ses périphériques. Maintenant je sais ! En fait, cet accessoire permettra à tous ceux qui désirent utiliser des fonctions du système, de voir comment se décomposent les commandes.

Exemple: *GetEof* fichier */Open* fichier */Read* fichier etc...

Notes sur le programme (ÓÓ)

Traceur est écrit avec l'assembleur Apw. Voici quelques notes sur les fichiers nécessaires pour le réassembler.

Le fichier LK est là pour assembler et linker le fichier Linkall.
Le fichier Linkall contient la liste des fichiers et equates.
Le fichier MK permet de générer le fichier de macros.
Le fichier RK permet de compiler les ressources.
Le fichier Main contient le module principal.
Fichier InitTools: sans commentaire!
Fichier Resource.S: sans commentaire.

Les deux routines principales du Traceur se trouvent dans le fichier Main, après le commentaire "Partie vitale du programme":

AdMyRoutine1. Cette routine traite les appels vers le vecteur $E1/00B0 dont les paramètres d'entrée sont :

Pushlong £AdresseParamètres
Pushword £Commande
Jsl $E1/00B0

Ceci est la méthode classique de GS/OS.

AdMyRoutine2. Cette routine traite les appels vers le vecteur $E1/00A8 dont les paramètres d'entrée sont :

Jsl $E1/00A8
dc i2'Commande'
dc i4'AdresseParamètre'

Ceci est la méthode classique de Prodos 16.

A l'entrée des deux routines, il y a analyse de la commande, calcul de l'adresse de ses paramètres, sélection du critère le plus important à afficher, puis l'affichage proprement dit.

Je vous souhaite de fructueux espionnages.

Reference:
GS/OS Reference Manual, d'Apple Computer.

```
********************(ÓÓÓ)**************
*       Partie vitale du programme
***********************************************

AdMyRoutine1    Anop
                Jmp >MyRoutineGsOs

;.................................

MyRoutineGsOs   Anop
                Phb
                Phd             >>>

;.................................

                Phk
                Plb

                Lda MyPageZero
                Tcd

;.................................

                Jsr AdParamGsOS
; <HandleBcontientmaintenant
; l'adressedesparamètres

                Lda AdCommande,s
                Sta <NumPonction
                Jsr SousProgB0P16

                Jsr UpdateWind2

                Pld             <<<
                Plb

RoutineGsOs     ds 4

;-------------------------------------

GsOs2007        Anop            * Inconnu
GsOs200D        Anop            * Null
GsOs200F        Anop            * GetSysPrefs
GsOs201B        Anop            * GetLevel
GsOs201D        Anop            * BeginSession
GsOs201E        Anop            * EndSession
GsOs201F        Anop            * SessionStatus
GsOs2021        Anop            * Inconnu
GsOs2022        Anop            * Inconnu
GsOs2023        Anop            * Inconnu
GsOs2026        Anop            *ResetCacheGsOs
GsOs2027        Anop            * GetName
GsOs2028        Anop            * GetBootVolume
GsOs202A        Anop            * GetVersion
GsOs2033        Anop            * Inconnu
GsOs2034        Anop            ? AddNotifyProc
GsOs2035        Anop            ? DelNotifyProc
GsOs2036        Anop            ? DRename
GsOs2037        Anop            ? GetStdRefNum
GsOs2038        Anop            ? GetRefNum
GsOs2039        Anop            ? GetRefInfo
GsOs203A        Anop            Sécurité
GsOs203B        Anop
GsOs203C        Anop
GsOs203D        Anop
GsOs203E        Anop
GsOs203F        Anop
                Stz NomFichier
                Rts

;---------------------  Create  -------------
;           0 1   ^^^^^^^^
; Create : 0200  55626100
;         PCount Pointeur
;

GsOs2001        Anop            Create
GsOs2002        Anop            Destroy
GsOs2003        Anop            OsShutDown
GsOs2004        Anop            ChangePath
```

```
GsOs2005        Anop        SetFileInfo
GsOs2006        Anop        GetFileInfo
GsOs2008        Anop        Volume
GsOs200B        Anop        ClearBackUp
GsOs200E        Anop        ExpandPath
GsOs2020        Anop        GetDevNumber
GsOs2024        Anop        Format
GsOs2025        Anop        EraseDisk
GsOs2029        Anop        Quit
                Ldy #2
                Lda [HandleB],Y
                Sta <Handle
                Iny
                Iny
                Lda [HandleB],Y
                Sta <Handle+2
                Brl MoveTitre

;----------------- GetPrefix --------------
;             0 1  ^^^^
; Get Prefix : 0200  0800    55626100
;            PCount PrefNum Pointeur

GsOs200A        Anop        GetPrefix
GsOs200C        Anop        SetSysPrefs
GsOs2011        Anop        NewLine
GsOs2012        Anop        Read
GsOs2013        Anop        Write
GsOs2014        Anop        Close
GsOs2015        Anop        Flush
GsOs2017        Anop        GetMark
GsOs2019        Anop        GetEof
GsOs201A        Anop        SetLevel
GsOs201C        Anop        GetDirEntry
GsOs202B        Anop        GetFstInfo
GsOs202C        Anop        DInfo
GsOs202D        Anop        DStatus
GsOs202E        Anop        DControl
GsOs2032        Anop        UnBindInt
                Ldy #2
                Lda [HandleB],Y
                Pha

MovePref        Anop
                PushLong #NumPref+3
                Pushword #4
                _Int2Hex
                PushLong #NumPref
                PushLong #NomFichier
                Pushlong #8
                _BlockMove
                Rts

;----------------------- Open ---------------
;         0 1    2 3   ^^^^^^^^
; Open : 0200   0100   55626100
;       PCount RefNum PointeurNom

GsOs2009        Anop        Set Prefix
GsOs2010        Anop
                Ldy #4
                Lda [HandleB],Y
                Sta <Handle
                Iny
                Iny
                Lda [HandleB],Y
                Sta <Handle+2
                Brl MoveTitre

;----------------- Set Mark ---------------
;           0 1    2 3    4 5 ^^^^^^^^
; SetMark : 0200   0100   0000 00100000
;         PCount RefNum Base Displacement

GsOs2016        Anop        SetMark
GsOs2018        Anop        SetEof
                Ldy #6

MoveNumBloc     Anop
                Lda [HandleB],Y
                Pha
                PushLong #NumBloc+3+4
                Pushword #4
                Iny
                Iny
                Lda [HandleB],Y
                Pha
                PushLong #NumBloc+3
                Pushword #4
                _Int2Hex
                _Int2Hex
                PushLong #NumBloc
                PushLong #NomFichier
                Pushlong #13
                _BlockMove
                Rts

;--------------- DRead ----------------------

; DRead:
;   0 1    2 3    4 5 6 7    8 9 A B ^^^^^^^^
; 0600   0200   00800200   00000000 00000000
; PCount RefNum Buffer     Buffer   StartBlock

GsOs202F        Anop        DRead
GsOs2030        Anop        DWrite
                Ldy #12
                Bra MoveNumBloc

;----------------- Set Mark ---------------
;           ^^^^^
; BindInt : 02 00  01 00  00 00 00 10 00 00
;         PCount IntNum vrn   IntCode
;
GsOs2031        Anop        BindInt
                Ldy #4
                Lda [HandleB],Y
                Pha
                Brl MovePref

;-----------------------------------------

AdMyRoutine2    Anop
                Jmp >MyRoutineP16

;------------------------------

MyRoutineP16    Anop
                Phb
                Phd             >>>

;------------------------------

                Phk
                Plb
                Lda MyPageZero
                Tcd
;------------------------------
```

```
                Jsr AdParamP16
; <HandleB contient maintenant
; l'adresse des paramètres

                Ldy #FonctP16
                Lda [Handle],Y
                Sta <NumFonction

                Jsr SousProgB0P16

                Jsr UpdateWind2

                Pld             <<<
                Plb

RoutineP16      ds 4

;------------------- Table P16 --------------
P160007         Anop            * GetEntry
P16000D         Anop            * Inconnu
P16000F         Anop            * Inconnu
P16001B         Anop            * GetLevel
P16001D         Anop            * Inconnu
P16001E         Anop            * Inconnu
P16001F         Anop            * Inconnu
P160026         Anop            * Inconnu
P160027         Anop            * GetName
P160028         Anop            * GetBootVolume
P16002A         Anop            * GetVersion
P16002B         Anop            * Inconnu
P16002D         Anop            * Inconnu
P16002E         Anop            * Inconnu
P16002F         Anop            * Inconnu
P160030         Anop            * Inconnu
P160033         Anop            Sécurité
P160034         Anop
P160035         Anop
                Stz NomFichier
                Rts

;------------------- Create P16 ------------
;               ^^^^^^^^
; Create (P16) : 55626100
;               Pointeur
P160001         Anop            Create
P160002         Anop            Destroy
P160003         Anop            Rename
P160004         Anop            ChangePath
P160005         Anop            SetFileInfo
P160006         Anop            GetFileInfo
P160008         Anop            Volume
P16000B         Anop            ClearBackUpBit
P16000C         Anop            WriteProtect
P16000E         Anop            ExpandPath
P160020         Anop            GetDeviceNum
P160024         Anop            Format
P160025         Anop            EraseDisk
P160029         Anop            Quit
                Lda [HandleB]
                Sta <Handle
                Ldy #2
                Lda [HandleB],Y
                Sta <Handle+2
                Brl MoveTitreP16

;---------------- Set Prefix P16 ------------
;                 0 1 ^^^^^^^^
; SetPrefix (P16) : 0000 55626100
;                   Num  Pointeur
P160009         Anop            SetPrefix
P160010         Anop            Open
P16002C         Anop            DInfo
                Ldy #2
                Lda [HandleB],Y
                Sta <Handle
                Iny
                Iny
                Lda [HandleB],Y
                Sta <Handle+2
                Brl MoveTitreP16

;---------------- Get Prefix P16 ------------
;                   ^^^^
; GetPrefix (P16) : 0000 55626100
;                   Num  Prefix
P16000A         Anop            GetPrefix
P160011         Anop            NewLine
P160012         Anop            Read
P160013         Anop            Write
P160014         Anop            Close
P160015         Anop            Flush
P160017         Anop            GetMark
P160019         Anop            GetEof
P16001A         Anop            SetLevel
P16001C         Anop            GetDirEntry
P160021         Anop            GetLastDev
P160032         Anop            DeallocInterrupt
                Lda [HandleB]
                Pha
                Brl MovePref

;---------------- Set_Mark P16 ------------
;                   0 1 ^^^^^^^^
; Set_Mark (P16) : 0000 00000000
;                   Num   Valeur
P160016         Anop            SetMark
P160018         Anop            SetEof
                Ldy #2
                Brl MoveNumBloc

;---------------- Read Block P16 ----------
;                                ^^^^^^^^^^^
;ReadBlock (P16):00 00  00 00 00 00 00 00 00 00
;               DevNum DataBuffer  Block Num
P160022         Anop            ReadBlock
P160023         Anop            WriteBlock
                Ldy #6
                Brl MoveNumBloc

;---------------- AllocInterrupt P16 -------
;                   0 1  ^^^^
; AllocInt (P16) : 0000  0000
;                   IntNum IntCode
P160031         Anop            AllocInterrupt
                Ldy #2
                Lda [HandleB],Y
                Pha
                Brl MovePref

;-------------- Affiche Résultat ------------
UpdateWind2     Anop
                LongResult
```

```
                    _GetPort
                    PushLong <WindTraceur
                    _SetPort
                    PushLong #RectErase
                    _EraseRect
                    Pushword #5
                    Pushword #9
                    _MoveTo
                    PushLong #AdresseDep
                    _DrawString
                    Pushword #295-4
                    Pushword #9
                    _MoveTo
                    PushLong <NomFonction
                    _DrawString
                    Lda NomFichier
                    Beq NoParamAux
                    Pushword #5
                    Pushword #0
                    _Move
                    PushLong #NomFichier
                    _DrawString
NoParamAux          Anop
                    PushLong #RectErase
                    PushLong #RectErase3
                    PushLong #2*4
                    _BlockMove
                    PushLong #RectErase3
                    _ValidRect
                    Lda <ValPause
                    Beq NoPause
                    Jsr Wait
NoPause             Anop
                    _SetPort
                    Rts
;------------  Boucle d'attente  -------------
Wait                Anop
                    Pushword #625
                    Pushword #9
                    _MoveTo
                    Pushword #PommeOuv
                    _DrawChar
                    Shortm
                    Sta >Strobe
LoopWait            Anop
                    Lda >Kbd
                    Bpl LoopWait
                    Sta >Strobe
                    Longm
                    PushLong #RectErase2
                    _EraseRect
                    Rts
;------  Création de la Table Ressources  -----
CalcMapRes          Anop
                    LongResult
                    Pushword #rPString
                    _CountResources
                    Pla
                    Plx
                    LongResult
                    LongResult
                    Phx
                    Pha
                    PushLong #4
                    _LongMul
                    Pla
                    Plx
                    Ply
                    Ply
; Crée la table index handle GSOS
                    Jsr NewHandleTR
                    Stx <IndexResGsOs+2
                    Sta <IndexResGsOs
                    Lda #ResCreate|-16
                    Ldx #ResCreate
                    Sta <HandleB+2
                    Stx <HandleB
                    Stz <Var1
                    Jsr LoopLoadResGs
                    Lda <HandleB
                    Sbc #ResCreate
                    Sta <NbResGsos
; Calcule le début de la table pour P16
; IndexResGsOs+<Var1
                    Clc
                    Lda <IndexResGsOs
                    Adc <Var1
                    Sta <IndexResP16
                    Lda <IndexResGsOs+2
                    Adc #0
                    Sta <IndexResP16+2
                    Lda #ResCreateP16|-16
                    Ldx #ResCreateP16
                    Sta <HandleB+2
                    Stx <HandleB
                    Jsr LoopLoadResGs
                    Lda <HandleB
                    Sbc #ResCreateP16
                    Clc
                    Adc <NbResGsos
                    Sta <NbResGsos
                    Rts
;---------------------  Detach Resource  ------
LoopLoadResGs       Anop
                    Jsr LoadRes
                    Bcs EndResGsOs
                    Ldy <Var1
                    Sta [IndexResGsOs],Y
                    Iny
                    Iny
                    Txa
                    Sta [IndexResGsOs],Y
                    Iny
                    Iny
                    Sty <Var1
```

```
                Pushword #rPString
                PushLong <HandleB
                _DetachResource
                Inc <HandleB
                Bra LoopLoadResGs
EndResGsOs      Anop
                Rts

;--------------- New Handle ----------------

NewHandleTR     Anop
                LongResult
                Phx             High
                Pha             Low
                Pei <MyUserID
;       FEDCBA9876543210
;       .... ...
                Pushword #%1100000000001000
                PushLong #0
                _NewHandle
                Pla
                Plx
                Bcc NewHandDeref
                Rts

;--------------- Load Resource ----------

LoadRes         Anop
                LongResult
                Pushword #rPString
                PushLong <HandleB
                _LoadResource
                Pla
                Plx
                Bcc NewHandDeref
                Rts

NewHandDeref    Anop
                Stx <Handle+2
                Sta <Handle

                Ldy #2
                Lda [Handle],Y
                Tax
                Lda [Handle]
                Rts

;-- Move Titre GsOs <Handle -> Nom de fichier

MoveTitre       Anop
                Lda [Handle]    Longueur du nom
                Inc <Handle     Début du nom
;                               (GsString)

MoveTitreCom    Anop
                Inc <Handle
                Cmp #49
                Bcc LongueurMax

                Lda #49

LongueurMax     Anop
                Sta NomFichier

                PushLong <Handle
                PushLong #NomFichier+1
                Pushword #0
                Pha             Longueur
                _BlockMove
                Rts

;-- Move Titre P16 <Handle -> Nom de fichier --

MoveTitreP16    Anop
                Lda [Handle]    Longueur du nom
                And #$00FF
                Bra MoveTitreCom

;--------- Calcul Nom de la fonction GsOs ---

NomFonctGsOs    Anop
;               Sec             Cette partie
;                               est déjà faite
;               Lda <NumFonction
;               Sbc #GsOsCreate
;               Asl A           *2
;               Asl A           *4
                Tay
                Lda [IndexResGsOs],Y
                Tax
                Iny
                Iny
                Lda [IndexResGsOs],Y
                Stx <NomFonction
                Sta <NomFonction+2
                Rts

;--------- Calcul Nom de la fonction P16 ---

NomFonctP16     Anop
;               Sec
;               Lda <NumFonction
;               Sbc #P16Create
;               Asl A           *2
;               Asl A           *4
                Tay
                Lda [IndexResP16],Y
                Tax
                Iny
                Iny
                Lda [IndexResP16],Y
                Stx <NomFonction
                Sta <NomFonction+2
                Rts

;------------------ Branchement GsOs/P16 ---

SousProgB0P16   Anop
                Cmp #GsOsCreate
                Bcc CommandeP16

                Sbc #GsOsCreate
                Asl A
                Pha             >>
                Jsr NomFonctGsOs
                Plx             <<
                Jsr (TableComGsOs,X)
                Rts

CommandeP16     Anop
                Sec
                Sbc #P16Create
                Asl A
                Pha             >>
                Jsr NomFonctP16
                Plx             <<
                Jsr (TableComP16,X)
                Rts

;--------- Calcul des paramètres de GsOs ----

AdParamGsOS     Anop
                Jsr AdresseRetour

                Lda AdParam+2+2,s Adresse
;                                 des paramètres
```

```
                Sta <HandleB+2
                Pha
                PushLong #AdrParm
                Pushword #2
                _Int2Hex
                Lda AdParam+2,s
                Sta <HandleB
                Pha
                PushLong #AdrParm2
                Pushword #4
                _Int2Hex
                Rts
;----------  Calcul des paramètres de P16  ----
AdParamP16      Anop
                Jsr AdresseRetour
                Ldy #ParamP16+2 Adresse
;                               Paramètre P16
                Lda [Handle],Y
                Sta <HandleB+2
                Pha
                PushLong #AdrParm
                Pushword #2
                _Int2Hex
                Ldy #ParamP16
                Lda [Handle],Y
                Sta <HandleB
                Pha
                PushLong #AdrParm2
                Pushword #4
                _Int2Hex
                Rts
;-------  Calcul Adresse de retour  -----------
AdresseRetour   Anop
                Lda AdRetour+2+2+2,s
                And #$00FF
                Sta <Handle+2  Uniquement
;                              pour P16
                Pha
                PushLong #AdrRet
                Pushword #2
                _Int2Hex
                Lda AdRetour+2+2,s
                Dea
                Dea
                Dea
                Sta <Handle
                Pha
                PushLong #AdrRet2
                Pushword #4
                _Int2Hex
                Rts
;-----------------------------------------------
TableComGsOs    Anop
                dc i'GsOs2001'      Create
[Voir table complète dans le fichier disque]
                dc i'GsOs203F'
TableComP16     Anop
                dc i'P160001'       Create
[Voir table complète dans le fichier disque]
                dc i'P160035'       Sécurité
                END
************************************************
```

Comment réassembler le Traceur

Tout d'abord, il vous faut impérativement APW ou Orca **version 1.1**, avec l'assembleur et Rez comme langages. Ensuite, voici comment procéder (APW n'est pas simple tous les jours):

1/ Regardez la date de vos fichiers APW: car il va falloir **débuguer APW** d'abord! Si ces dates commencent au 1/4/90, vous avez la dernière version.

Dans ce cas, éditez le fichier **M16.Window**. Cherchez la ligne suivante:

owRefCon GEQU 178, et remplacez-la par:
owrRefCon GEQU 178

2/ Si ces dates sont antérieures au 1/4/90, vous avez la version 1.1 "ancienne". Dans ce cas, il faut remplacer vos fichiers d'equates et de macros pour l'assembleur **E16.GSOS, E16.RESOURCES, et M16.RESOURCES** par ceux de /TOOLBOX.MAG.03/ TRACEUR.NDA/ LIBRAIRIES. Il manquait beaucoup d'equates dans la version ancienne, et certains fichiers contenaient des bizarreries. En revanche, M16.Window n'a pas besoin d'être corrigé.

3/ Pour les deux versions : copiez le fichier **TYPES.RES.AUX** depuis le sous-catalogue /TOOLBOX.MAG.03/ TRACEUR.NDA/ LIBRAIRIES dans le dossier d'APW où vous avez mis vos Includes pour Rez. Copiez ensuite le fichier **M16.A** depuis le sous-catalogue /TOOLBOX.MAG.03/ TRACEUR.NDA/ LIBRAIRIES dans le dossier d'APW où vous avez mis vos Includes pour l'Assembleur. **Attention: placez ce fichier avant tous les autres fichiers "M16.XXX".**

4/ Ici se termine le débuguage d'APW. Reste à réassembler le Traceur. Cela se fait en trois temps:

a) Copiez sur le disque de votre choix, depuis le catalogue /TOOLBOX.MAG.03/ TRACEUR.NDA/ SOURCES les fichiers MK, RK, LK, LINKALL, MAIN, INITTOOLS, et RESOURCE.S (à l'exclusion des autres).

b) Editez le fichier **MK**: vous verrez trois lignes différentes qui fixent le **Préfixe 2.** Remplissez-les de façon qu'APW puisse trouver où vous avez mis vos Includes et Macros pour l'Assembleur, et vos Includes pour Rez.

c) Lancez le fichier **MK... Ouf!**

JYB

Riches, et contents !

(petit vade-mecum du rédacteur GS)

J.Y. Bourdin (♂)

On ne peut rien cacher à un Rédacteur en Chef qui lit articles, programmes et courrier (et encore moins à une maquettiste - merci Dominique) de ses mauvaises habitudes en matière de rédaction (surtout quand il a lui-même tant de mal à s'en libérer: ToolBox Mag est parsemé des "fautes" que je relèverai ici).

Je ne parle pas d'orthographe ou de français, mais de **nos mauvaises habitudes informatiques**, prises sur l'Apple II. Avec le GS, ordinateur graphique, et ses fontes proportionnelles, avec l'accès à toutes sortes d'imprimantes, il nous faut changer d'habitudes: il faut "**jeter AppleWorks 8**".

Jeter AppleWorks 8

Il est temps aujourd'hui de cesser d'utiliser des programmes non conformes aux standards GS: aussi bien, donc, GS Write que Gribouille. Que Madame Hodé me pardonne, mais ses arguments, pertinents en leur temps, contre l'utilisation des outils GS, ne sont plus de mise avec le niveau de maturité atteint aujourd'hui par le GS et ses systèmes.

Rédiger sur GS, c'est aujourd'hui, obligatoirement, utiliser les fontes GS: le mode texte n'est plus bon que pour les brouillons. Nous voyons déjà poindre l'étape suivante: l'utilisation d'effets PostScript dans la rédaction.

Je ne dis pas cela pour faire l'éloge de tel ou tel logiciel en particulier, comme AppleWorks GS, Beagle-Write ou Graphic Writer III: c'est d'abord d'Apple que je fais l'éloge. L'outil TextEdit de son système permet aujourd'hui à de simples (et petits) accessoires de bureau de faire du traitement de textes en couleurs, avec les fontes GS et leurs effets, à coût très faible aussi bien financièrement que du point de vue de l'encombrement disque, ou de l'occupation mémoire. **TextEdit périme AppleWorks 8** (même avec SuperFonts), il périme Gribouille, c'est ainsi.

Lecteurs, excusez-moi d'insister un peu: on ne peut pas conserver "provisoirement" nos vieux programmes: car **ils produisent des fichiers qui, au niveau même des données en Ascii brut, sont incompatibles avec les fichiers GS/Mac.** Quand j'ai publié dans Pom's une macro pour la conversion des fichiers traitement de texte AppleWorks 8 comme "cadeau d'adieu", il s'agissait simultanément d'un adieu à Pom's et d'un adieu à AppleWorks 8... Il n'est plus acceptable, pour un utilisateur de GS, de lire sur l'écran (et dans le fichier lui-même) « ^a » pour « â », et de ne pouvoir écrire « {é} » ou « √2 ≤ ∏ » qu'avec les «<X2>Zù<X3>b%<X1>» déments de SuperFonts.

Bien, mais cesser d'utiliser des logiciels périmés ne suffit pas. Il faut encore apprendre le nouveau clavier, puisque nous avons désormais 128 caractères de plus à notre disposition!

Les deux claviers

L'essentiel de cet article est écrit pour présenter le tableau des codes Ascii et de leur accès par le clavier du GS tel que nous les offrent les fontes GS, d'une part, le choix "Translation: Français" du GS/OS 5.02 français d'Apple d'autre part, et enfin la reconfiguration que nous propose Yvan Kœnig pour le 5.02 français (celle qu'on obtient avec les fichiers FrInit et Sys.Resources de la disquette ToolBox Mag 2, catalogue /Derniere.Heure).

Pourquoi préférer la configuration Kœnig à celle d'Apple? Il y a au moins deux raisons: d'une part, TextEdit n'accepte pas les combinaisons incluant la touche Control. Il y a donc certains caractères qui sont inaccessibles sous l'éditeur de TML Pascal II, par exemple, avec la configuration Apple.

D'autre part, le principe de la touche morte (l'accent, puis la lettre), qui est étendu par Yvan Kœnig aux majuscules accentuées, est beaucoup plus normal et évident pour un français. La configuration d'Apple a été concoctée pour nous... en Californie, et ce genre de choses se fait beaucoup mieux sur place, près des utilisateurs réels.

[Il est à noter qu'il y aurait une bonne raison de reprendre complètement cette configuration sur Mac: les Macs jusqu'au Mac Plus inclus n'ont pas de touche Control sur leur clavier, ce qui leur interdit purement et simplement l'accès par le clavier à certains de ces caractères (à partir du 248 par exemple). Autrement dit, des centaines de milliers d'utilisateurs de Macintosh ont accepté sans rien dire, pendant des années, de n'avoir aucun accès du tout par le clavier à certains codes Ascii. "Faites du copier-coller", leur disait-on, et ils cliquaient fièrement sur leurs souris! Si vous vous demandez si nous n'y allons pas un peu fort, utilisateurs de GS, à tant rouspéter, regardez ce qui se passe chez ceux qui ne rouspètent pas.]

Laser: le bug

Si vous utilisez une LaserWriter, vous vous apercevrez que le driver LaserWriter Apple est affecté d'un bug gênant: 16 caractères (ceux qui apparaissent en grisé dans le tableau pages 18/19, dont d'ailleurs la elle-même), ne sont purement et simplement pas imprimés!

Ces 16 caractères existent bien dans les fontes écran,

mais ils n'existent pas dans les fontes texte Adobe qui sont dans la mémoire de la LaserWriter. Il faudrait donc que le driver aille chercher ces caractères dans la fonte Symbol, où ils sont, et fasse l'échange avant impression.

Cela fait partie, selon le LaserWriter Reference, du cahier des charges de ce driver, et celui du Mac s'y soumet. On comprend cependant Apple de ne pas avoir rectifié ce bug qui date de la première version: il y faudrait une dizaine de lignes de PostScript, une bonne demi-heure de travail, et les heures perdues par les utilisateurs à comprendre, puis à contourner ce bug, ne doivent pas dépasser quelques centaines...

Pour le contourner, une seule solution: **remplacer soi-même, à la main, ces 16 caractères par leurs correspondants dans la fonte Symbol.** Bien entendu, en vertu de l'adage de Racine "Tout m'afflige, et me nuit, et conspire à me nuire", la plupart de ces caractères ne sont pas, dans la fonte Symbol, rangés à la même place que dans nos fontes écran, et s'obtiennent avec une combinaison clavier différente! Et bien entendu, en vertu du même adage, quelques-uns ne s'obtiennent pas de la même façon avec la configuration Apple et avec la configuration Kœnig!

Le tableau pages 18/19 vous indique donc comment obtenir, sur LaserWriter, ces caractères dans la fonte Symbol: les signes « Σ: » sont là pour cela. Par exemple, la dernière ligne de la page 18 se lit de la façon suivante:

« Le symbole « ∫ », de code Ascii $BA en hexadécimal et 186 en décimal, qui représente une intégrale, s'obtient en tapant « Option-B » avec la configuration Apple, et même chose avec la configuration Kœnig. Mais si vous voulez l'imprimer sur LaserWriter, il faut le remplacer par le symbole équivalent dans la fonte Symbol, lequel s'obtient par « Option-y » dans la configuration Apple, et par « Option-apostrophe puis U » dans la configuration Kœnig ». Evident, non?

Pour les utilisateurs de LaserWriter sur GS, je recommande une précaution: éditez une copie de vos fontes-écran Laser (Times, Helvetica, etc), et remplacez les caractères en question par un "pense-bête" pour vous rappeler de passer en Symbol.

Prendre les bonnes habitudes

Reste à utiliser toutes ces nouveautés. Et pour cela, à prendre de nouvelles habitudes.

Les tabulations

Est-ce un souvenir du vieil AppleWorks 1.4, qui n'était pas net avec les Tabs, ou une mauvaise habitude du débutant sur clavier? En tout cas, beaucoup d'entre nous (votre serviteur en premier) ont encore tendance, quand ils veulent aligner des colonnes, ou simplement commencer un paragraphe, à remplir avec des espaces, bien que tous les manuels de dactylographie, même sur d'antiques machines à écrire mécaniques, enseignent l'usage des tabulations.

Seulement, **avec nos fontes graphiques proportionnelles, ni les espaces ni les lettres n'ont la même largeur d'une fonte à l'autre**: quand vous remplissez avec des espaces, d'abord vous n'obtenez jamais un véritable alignement, ensuite vous vous condamnez à ne plus jamais changer de fonte. A quoi sert alors d'avoir un GS?

Apprenons donc à nous servir des "règles de tabulation" de nos logiciels, et acceptons que la touche Tab serve... aux tabulations!

Ponctuation et "espace dur"

Ce point d'exclamation, là, il est vraiment étroit! Si je mets un espace avant, ça sera plus lisible. Seulement, quand j'imprime, j'obtiens le résultat que vous voyez !

Pourquoi? Eh bien, l'espace (ce bon vieux code Ascii 32, $20), ce n'est pas un espace: c'est très exactement un "inter-mots", **un marqueur de séparation entre les mots.** C'est sur cet inter-mots que va jouer le logiciel quand vous lui demandez une justification totale: il va aligner des deux côtés en augmentant légèrement ces "inter-mots". **C'est aussi sur ces inter-mots qu'il va se repérer pour faire ses coupures de fin de ligne**: il coupe là où vous avez mis ces faux "espaces". Il considère ainsi comme parfaitement légal de couper entre le mot et le signe de ponctuation qui le suit.

Attention donc aux points, virgules, points-virgules, deux-points, etc...: mettez un espace après la ponctuation, mais n'en mettez pas avant. Laissez-les collés aux mots qui précèdent, ou alors... mettez un **vrai espace**: un "espace dur".

L'espace dur, ou "vrai espace", ou "espace collant", c'est celui dont vous vous servez pour écrire par exemple "ToolBox Mag": pour éviter la coupure entre "ToolBox" et "Mag", vous les reliez par un signe que vos logiciels considèrent comme un caractère, pas comme un intervalle. Dans ce cas-là, on tape "**Option-Espace**", la barre d'espace étant réservée à "je veux une coupure entre les mots à cet endroit".

Une note aussi sur les **points de suspension**: beaucoup d'entre nous tapent souvent quelques points, de trois à cinq en moyenne, pour marquer les points de suspension. Seulement, certaines fontes ImageWriter intelligentes, ont une ponctuation spécialement étudiée, qui permet d'éviter de mettre un espace dur avant la ponctuation: le caractère lui-même a été dessiné en réservant du blanc sur la gauche (à droite, il faut mettre un "inter-mots"). Conséquence: on voit parfois des points de suspension qui donnent ceci . . . . .

[Suite de l'article page 20]

Accès aux caractères étendus sur le clavier de l'Apple IIGS

(Version Apple GS/Mac et version Yvan Koenig)

Carac	Hexa	Déci	Nom PostScript du caractère	Apple	Koenig
#	$23	035	numbersign (dièse)	Option £	Id
@	$40	064	at (arobas ou a commercial)	Option` (accent grave)	Option ` + à
[	$5B	091	bracketleft (crochet gauche)	Option 5	Id
\	$5C	092	backslash (barre oblique à droite)	Option /	Id
]	$5D	093	bracketright (crochet droit)	Option ° (degré)	Id
{	$7B	123	braceleft (accolade gauche)	Option (	Id
|	$7C	124	bar (barre verticale)	Option L	Id
}	$7D	125	braceright (accolade droite)	Option)	Id
~	$7E	126	asciitilde	Option n	Option n + Espace
Ä	$80	128	Adieresis (A tréma)	Option Control-Z	¨ + A
Å	$81	129	Aring (A couronné)	Option Z	Id
Ç	$82	130	Ccedilla (C cédille)	Option ç	Id
É	$83	131	Eacute (E accent aigu)	Option Control-E	Option ' + E
Ñ	$84	132	Ntilde	Option Control-N	Option n + N
Ö	$85	133	Odieresis (O tréma)	Option Control-T	¨ + O
Ü	$86	134	Udieresis (U tréma)	Option Control-U	¨ + U
á	$87	135	aacute (a accent aigu)	Option Control-W	Option ' + a
à	$88	136	agrave (a accent grave)	à	Id
â	$89	137	acircumflex (a accent circonflexe)	Option Control-C	^ + a
ä	$8A	138	adieresis (a tréma)	Option Control-S	¨ + a
ã	$8B	139	a tilde	Option Control-Q	Option n + a
å	$8C	140	aring (a couronné)	Option Control-X	Option 8
ç	$8D	141	ccedilla (c cédille)	ç	Id
é	$8E	142	eacute (e accent aigu)	é	Id
è	$8F	143	egrave (e accent grave)	è	Id
ê	$90	144	ecircumflex (e accent circonflexe)	Option e	Id
ë	$91	145	edieresis (e tréma)	Option é	Id
í	$92	146	iacute (i accent aigu)	Option Control-I	Option ' + i
ì	$93	147	igrave (i accent grave)	Option Control-K	Option ` + i
î	$94	148	icircumflex (i accent circonflexe)	Option i	Id
ï	$95	149	idieresis (i tréma)	Option I	Id
ñ	$96	150	ntilde	Option Control-B	Option n + n
ó	$97	151	oacute (o accent aigu)	Option Control-P	Option ' + o
ò	$98	152	ograve (o accent grave)	Option Control-M	Option ` + o
ô	$99	153	ocircumflex (o accent circonflexe)	Option ^	^ + o
ö	$9A	154	odieresis (o tréma)	Option Control-G	¨ + o
õ	$9B	155	otilde	Option Control-L	Option n + o
ú	$9C	156	uacute (u accent aigu)	Option Control-R	Option ' + u
ù	$9D	157	ugrave (u accent grave)	ù	Id
û	$9E	158	ucircumflex (u accent circonflexe)	Option Control-F	^ + u
ü	$9F	159	udieresis (u tréma)	Option Control-J	¨ + u
†	$A0	160	dagger (dague)	Option t	Id
°	$A1	161	degree (degré)	° (degré)	Id
¢	$A2	162	cent	Option C	Id
£	$A3	163	sterling	£	Id
§	$A4	164	section	§	Id
•	$A5	165	bullet (balle)	Option , (point)	Id
¶	$A6	166	paragraph	Option §	Id
ß	$A7	167	germandbls (double s allemand)	Option b	Id
®	$A8	168	registered	Option r	Id
©	$A9	169	copyright	Option c	Id
™	$AA	170	trademark	Option T	Id
´	$AB	171	acute (accent aigu)	Option 1	Option ' + Espace
¨	$AC	172	dieresis (tréma)	¨ (tréma)	¨ + Espace
≠	$AD	173	notequal (non-égal)	Option = / Σ: Option p	Id
Æ	$AE	174	AE	Option A	Id
Ø	$AF	175	0slash (zéro barré)	Option 0 (zéro)	Id
∞	$B0	176	infinity	Option , / Σ: Option .	Id
±	$B1	177	plusminus (plus ou moins)	Option + / Σ: id	Id
≤	$B2	178	lessequal (inférieur ou égal)	Option < / Σ: id	Id
≥	$B3	179	greaterequal (supérieur ou égal)	Option > / Σ: id	Id
¥	$B4	180	yen	Option Control-Y	Option Y
µ	$B5	181	mu	Option m	Id
∂	$B6	182	partialdiff (différentielle partielle)	Option d / Σ: id	Id
∑	$B7	183	summation (somme)	Option S / Σ: Option z	Id / Σ: ^ + A
∏	$B8	184	product (produit - PI majuscule)	Option P / Σ: P	Id
π	$B9	185	pi (pi minuscule)	Option p / Σ: p	Id
∫	$BA	186	integral (intégrale)	Option B /Σ: Option y	Id/Σ: Option ' + U

ª	$BB	187	ordfeminine (ordinal féminin)	Option U	Id
º	$BC	188	ordmaculine (ordinal masculin)	Option u	Id
Ω	$BD	189	Omega	Option Q / Σ: Opt W	Id
æ	$BE	190	ae	Option a	Id
ø	$BF	191	oslash (zéro barré minuscule)	Option à	Id
¿	$C0	192	questiondown (interrogation renversé)	Option ?	Id
¡	$C1	193	exclamdown (exclamation renversé)	Option !	Id
¬	$C2	194	logicalnot (non logique)	Option l	Id
√	$C3	195	radical (racine)	Option V / Σ: Option :	Id
ƒ	$C4	196	florin	Option f	Id
≈	$C5	197	approxequal (approximativt égal)	Option x/ Σ: Option U	Id
Δ	$C6	198	Delta	Option D / Σ: D	Id
«	$C7	199	guillemotleft (guillemets gauches)	Option è	Id
»	$C8	200	guillemotright (guillemets droits)	Option 7	Id
…	$C9	201	ellipsis (points de suspension)	Option ;	Id
	$CA	202	blank (espace dur)	Option Espace	Id
À	$CB	203	Agrave (A accent grave)	Option Control-à	Option ` + A
Ã	$CC	204	Atilde	Option Control-A	Option n + A
Õ	$CD	205	Otilde	Option Control-O	Option n + O
Œ	$CE	206	OE	Option O	Id
œ	$CF	207	oe	Option o	Id
–	$D0	208	endash (trait d'union court)	Option _ (souligné)	Id
—	$D1	209	emdash (trait d'union long)	Option - (tiret)	Id
“	$D2	210	quotedblleft (guillemets typo gches)	Option "	Id
”	$D3	211	quotedblright (guillemets typo droits)	Option 3	Id
‘	$D4	212	quoteleft (guilm typo simples gches)	Option '(apostr)	Option 1
’	$D5	213	quoteright (guilm typo simples droits)	Option 4	Id
÷	$D6	214	divide (division)	Option :	Id
◊	$D7	215	lozenge (losange)	Option v / Σ: Option q	Id
ÿ	$D8	216	ydieresis (y tréma)	Option Control-H	¨ + y
Ÿ	$D9	217	Ydieresis (Ytréma)	Option Y	¨ + Y
⁄	$DA	218	fraction	Option X	Id
¤	$DB	219	currency (monnaie)	Option R	Id
‹	$DC	220	guilsinglleft (guillemet simple gche)	Option w	Id
›	$DD	221	guilsinglright (guillemet simple droit)	Option W	Id
fi	$DE	222	fi	Option g	Option g
fl	$DF	223	fl	Option G	Id
‡	$E0	224	daggerdbl (dague double)	Option q	Id
·	$E1	225	periodcentered (point centré)	Option *	Id
‚	$E2	226	quotesinglbase (guillemet simple bas)	Option $	Id
„	$E3	227	quotedblbase (guillemets doubles bas)	Option 2	Id
‰	$E4	228	perthousand (pour mille)	Option %	Id
Â	$E5	229	Acircumflex (A accent circonflexe)	Option z	^ + A
Ê	$E6	230	Ecircumflex (E accent circonflexe)	Option E	^ + E
Á	$E7	231	Aacute (A accent aigu)	Option 9	Option ' + A
Ë	$E8	232	Edieresis (E tréma)	Option K	¨ + E
È	$E9	233	Egrave (E accent grave)	Option k	Option ` + E
Í	$EA	234	Iacute (I accent aigu)	Option J	Option ' + I
Î	$EB	235	Icircumflex (I accent circonflexe)	Option H	^ + I
Ï	$EC	236	Idieresis (I tréma)	Option j	¨ + I
Ì	$ED	237	Igrave (I accent grave)	Option h	Option ` + I
Ó	$EE	238	Oacute (O accent aigu)	Option M	Option ' + O
Ô	$EF	239	Ocircumflex (O accent circonflexe)	Option ¨ (tréma)	^ + O
	$F0	240	apple (Pomme…)	Option & / Σ: id	Id
Ò	$F1	241	Ograve (O accent grave)	Option s	Option ` + O
Ú	$F2	242	Uacute (U accent aigu)	Option y	Option ' + U
Û	$F3	243	Ucircumflex (U accent circonflexe)	Option 8	^ + U
Ù	$F4	244	Ugrave (U accent grave)	Option ù	Option ` + U
ı	$F5	245	dotlessi (i sans point)	Option N	Id
ˆ	$F6	246	circumflex (accent circonflexe)	Option 6	Id
˜	$F7	247	tilde	Option Control-^	Option 9
¯	$F8	248	macron	Option Control-_	Option E
˘	$F9	249	breve (brève)	Option Control-V	Option H
˙	$FA	250	dotaccent (accent-point)	Option F	Id
˚	$FB	251	ring (couronne)	Option Escape	Option J
¸	$FC	252	cedilla (cédille)	Option Control-ç	Option K
˝	$FD	253	hungarumlaut (umlaut hongrois)	Option Control-§	Option M
˛	$FE	254	ogonek	Option Control-D	Option h
ˇ	$FF	255	caron	Option DEL	Id

Accès aux caractères étendus sur le clavier de l'Apple II GS

(Version Apple GS/Mac et version Yvan Koenig)

[Suite de la page 17]

Or, **il y a un caractère "points de suspension" dans nos nouveaux caractères**: il est obtenu par "Option point-virgule", et il donne ceci...

Le retour des anciens

Quand on découvre tous ces nouveaux caractères, on se limite souvent à regarder au-dessus de 128: effectivement, là, c'est l'abondance. On en arrive à oublier ces caractères que nous avions appris à considérer comme des caractères "américains". Ces bonnes vieilles {accolades}, ces bons vieux [crochets], que nous avions sacrifiés pour nos é, nos è et nos §, les voici de retour, en même temps que nos é et nos è.

On peut même maintenant écrire des sources C imprimables sans avoir besoin de passer en américain: C facile, comme dit quelqu'un. Plus besoin de choisir entre ù et |, jusqu'au # qui est enfin redevenu un #, et qui ne se confond plus avec la £. Tout cela sur un GS 100% francisé, écran français, clavier français, translation française.

Apostrophes et guillemets

Amis qui écrivez sur votre GS, on voit en cinq lignes si vous êtes dans le coup du GS ou si vous êtes un "has-been du 8 bits": **les "vieux" écrivent "ceci", là où les "jeunes" écrivent "ceci".**

Plus grave encore, vous vous créez un problème: comme les "guillemets typographiques" (d'imprimerie) sont de deux sortes, gauches et droits, le remplacement des "guillemets dactylographiques" (de machine à écrire) n'est pas évident à faire automatiquement.

Ceci dit, si vous étudiez la liste des types de guillemets typographiques dont nous disposons, c'est vraiment l'abondance. Je crois savoir que les «guillemets doubles classiques» sont réservés à des citations textuelles, les "guillemets doubles hauts" à des citations sans nom d'auteur, les 'guillemets-apostrophes' à des références à un terme familier.

Mais je ne connais guère l'usage, je l'avoue, des ‹guillemets simples›, ni de ces curieux guillemets bas (droits seulement: , et „), que je distingue avec peine d'une virgule, simple ou double. Si vous avez des tuyaux à ce sujet, merci d'en faire profiter les autres...

Le guillemet-apostrophe droit (') est aussi parfois appelé "apostrophe typographique": il est effectivement utile, avec certaines fontes, de remplacer toutes les apostrophes ordinaires par cette apostrophe: mais ce n'est pas toujours le cas. Seul compte le résultat papier: l'apostrophe doit être inclinée à gauche, mais pas trop. Certains logiciels vous proposent d'ailleurs de remplacer l'apostrophe ordinaire par l'accent aigu (´). Il me semble difficile de chercher un standard: je conserve donc, par défaut, la vieille apostrophe.

♦ et ◊

C'est sans doute la faute préférée de ToolBox Mag, sinon la plus fréquente, du moins la plus visible: confondre ♦ et ◊. C'est sur le IIe/IIc qu'il y a une touche ♦, le GS n'a que ◊ et Option.

- Mais, dans mes fontes, je n'ai que la ♦, comment fais-tu pour vider ta Pomme?

- Je m'en doutais, bien peu (y compris sur Mac) pratiquent ce truc pourtant évident, parfaitement normal pour qui écrit sur GS. Les ◊ de cet article sont engendrées normalement, sans aucun trucage, avec la fonte Symbol (oui, pour la Laser, voir ci-dessus). Vous donnez votre langue au chat?

Déroulez le menu "Style"; essayez le style "Silhouette" ou "Eclairé" (Outline): il se charge très bien de vider la Pomme! Pensez simplement à prendre une taille de fonte un peu plus petite que le reste de votre ligne: le style "Outline" augmente la taille du caractère.

Les Crésus de la typographie...

En conclusion, je le reconnais: tout ce qui précède fait un peu donneur de leçons, un peu "prof". Pour ma défense (outre le fait que cette déformation n'est pas forcément la tare la plus indigne, après tout), je dirai que c'est d'abord à moi-même que je veux donner des leçons: en alertant les lecteurs sur nos fautes favorites, je sais qu'ils vont tiquer sur celles de ToolBox Mag, ce qui nous forcera à mieux faire.

Et puis, j'ai une autre raison: pour ne rien vous cacher, le GS, je suis pour. Or, tirer le maximum du GS, c'est apprendre la ToolBox, les Ressources, et tout ça, d'accord. Mais c'est aussi **se mettre au niveau de sa machine dans les utilisations les plus quotidiennes et les plus ordinaires, comme la rédaction.**

Je connais des experts en programmation du Resource Manager sous trois niveaux d'interruptions, qui utilisent pourtant toujours les vieux guillemets (et c'est vrai, les éditeurs de texte des logiciels de programmation sont globalement à la traîne de ce point de vue, accrochés qu'ils sont au vieux mode texte, même s'ils utilisent l'écran graphique).

Changer d'habitudes n'est facile pour personne, surtout quand, comme les habitudes d'écriture, elles sont devenues automatiques et inconscientes.

Mais il faut nous y faire: **nous sommes devenus riches!** Finies donc les "bidouilles de pauvres" obligés de tricher pour avoir les accents français, finies les habitudes des antiques machines à écrire mécaniques. Il faut **gérer l'opulence typographique**.

Avec toutes nos fontes, tous leurs styles, et tous leurs caractères, nous sommes devenus des **Crésus de la typographie**: d'accord, pas besoin d'en rajouter, cela fait "nouveaux riches". Mais pas besoin non plus de faire des complexes: **riches, et contents !**

Etude hard (🍎)

L'Apple II GS et la vidéo

Jacques Liautard

Vous avez un ordinateur Graphique, un GS. Ses images sont magnifiques sur votre écran couleur. Mais vous voyez aussi de belles images couleur ailleurs : sur votre écran TV par exemple. Forcément, vous avez envie de capter sur votre GS toutes ces images. Vous avez aussi envie d'envoyer sur la TV les images du GS, d'autant que la TV elle-même vous montre que cela se fait…

Les interfaces

Dans un monde idéal, il suffirait de brancher. Dans ce bas-monde, cela ne suffit pas. Le monde de l'image, de la lumière (comme celui du son d'ailleurs) est un monde où tout est continu, où l'on passe progressivement, insensiblement, d'une couleur à une autre, par exemple. Le monde du GS, celui de l'électronique, ne connait que les 0 et les 1, il est discontinu, "digital".
Le monde du digital est en train de rattraper l'autre : si je découpe un signal continu en tranches discontinues très, très fines et en très, très grand nombre, et que je reproduis ensuite ces tranches discontinues, le résultat est le même pour nos yeux et pour nos oreilles (24 images discontinues par seconde sont égales **à nos yeux** à du mouvement continu, c'est le principe du cinéma). Ce résultat est même meilleur, car ce qui est digital est d'une fiabilité absolue. Pour le son, ce processus arrive aujourd'hui à son terme : un lecteur de compact-disques se branche directement sur le GS.
Pour l'image, nous n'y sommes pas encore : d'autant que les réalités de la concurrence internationale ont multiplié les standards vidéo. Il faut donc des **"interfaces"**. Cet article voudrait montrer quels interfaces on peut utiliser, et comment, pour **insérer le GS dans la chaîne des images**. Car ça marche, et ça vaut le coup !

Distinguons

Mais il faut commencer par distinguer, car vous pouvez vouloir plusieurs choses différentes :

❶ **transporter dans le GS des images vidéo venant d'une source extérieure**. Cela s'appelle de la **digitalisation d'images**, et il vous faut pour cela une carte de digitalisation d'images, cette carte étant dans un des slots du IIGS.

❷ **enregistrer sur un magnétoscope les images affichées sur le moniteur du GS**, à la place d'une image vidéo. C'est assez facile, un simple câble y pourvoira pour le noir et blanc, un boîtier de conversion sera nécessaire pour la couleur.

❸ **superposer l'image de l'écran GS à une image vidéo de TV ou de magnétoscope**. Cela s'appelle faire de l'**incrustation**, et il y faut une carte spécifique.

❹ **décalquer à la main une image vidéo pour en faire une image GS** : idée curieuse, me direz-vous, pourquoi ne pas digitaliser (première fonction) ? Eh bien, il se trouve que la carte qui permet la troisième fonction (l'incrustation), permet aussi cette quatrième fonction, remplaçant ainsi la tablette graphique. Voyez le quatrième chapitre de cet article.

Chapitre 1 : L'acquisition d'images pour le GS (digitalisation)

Dans un premier temps, donc, nous supposerons que vous voulez capturer en fichiers Paint toutes les images que vous voyez autour de vous : c'est possible, bien sûr, cela s'appelle "digitaliser".

Digitaliser quoi ?

Tout : le GS peut digitaliser **n'importe quelle image vidéo**. Cela peut aller d'une image télévision (en reliant la sortie Péritel de votre poste à l'entrée vidéo de la carte de digitalisation), à une image de camescope, en passant par celle d'un magnétoscope, vidéo disque ou appareil photo vidéo (en reliant la sortie vidéo de l'appareil à l'entrée vidéo de la carte de digitalisation).

Les différents standards vidéo

NTSC : la norme américaine, totalement incompatible avec les autres normes.
SECAM : la norme française,
PAL : la norme des autres,
S-VHS et HI8 : pour le moment, aucun périphérique du IIGS ne reconnait ces signaux, mais une cassette enregistrée en S-VHS peut être relue au choix en Pal ou en Sécam.

Je dis bien "tout" : car tout ce qui se voit peut être transformé en image vidéo, il suffit de l'éclairer et de le filmer. On peut de cette façon digitaliser une photo papier, par exemple, et remplacer tous les scanners, Quickie ou autres : filmez et digitalisez.

Digitaliser, pour quoi faire ?

- vos lettres à en-tête.
- vos cartes de vœux.
- votre adresse au dos des lettres.
- accompagner l'adresse de vos amis sur les lettres avec leur photo.
- récupérer des images que vous ne pourriez pas dessiner vous-même.
- écrire un jeu d'aventures ou d'arcade (Space Ace) à base d'images digitalisées.
- etc, etc...

La seule limite est celle de votre imagination : d'autant qu'une fois votre image digitalisée, vous pouvez la retravailler avec votre logiciel de dessin habituel...

L'équipement : la carte de digitalisation

Pour transformer une image vidéo en image GS, il faut la digitaliser. Il faut donc acheter une carte de digitalisation d'images; et là, il s'agit de bien faire attention à ne pas se tromper au niveau du standard vidéo, car vous avez **deux choix** possibles : une carte en Pal ou une carte en NTSC.

Les cartes de digitalisation NTSC

Si vous achetez une carte en NTSC (**Computer Eyes** 195$, ou **Visionary Card** 349$, plus frais de port et de douane pour les deux), vous aurez la possibilité de digitaliser en noir et blanc ou encore en couleurs avec des tons pastels proches de la réalité. **Mais attention** : il faudra que vous possédiez également une **source vidéo en NTSC**, telle que magnétoscope, vidéodisque, camescope, caméra, ou encore appareil photo vidéo. Si vous essayez de digitaliser avec une source en Pal ou en Sécam vers une carte de digitalisation NTSC, vous n'aurez aucune image.

Tous ces matériels en NTSC ne se trouvent qu'aux USA ou en France dans des magasins spécialisés. Attention également : si vous achetez un magnétoscope NTSC, vous ne pourrez pas enregistrer les chaines françaises ou européennes, et vous ne pourrez lire une cassette que sur un téléviseur acceptant le NTSC. Il existe certes des **transcodeurs** Pal/Sécam/NTSC, mais leur prix est d'environ 8.000F!

[NDLR : à notre connaissance, l'achat d'une carte de digitalisation NTSC, en France, ne se justifie vraiment que pour qui veut digitaliser des images fixes en noir et blanc. Une caméra de surveillance noir et blanc, bon marché, dédiée à la digitalisation, peut faire alors un équipement correct, le moniteur du GS étant chargé de la visualisation.]

Les cartes de digitalisation en Pal

Si vous achetez une carte en Pal vous avez deux possibilités : la carte **Computer Eyes-Pal** (2.500 F environ), ou la carte **DGA-1** (5.000 F).

La carte Computer Eyes-Pal

C'est la moins chère, mais il faut compter 6 secondes minimum pour digitaliser une image. Ce qui veut dire que si vous essayer de digitaliser une image en mouvement, vous aurez aussi le mouvement, donc une image floue. Pour digitaliser sans problème avec la Computer Eyes, il faut partir d'une **image fixe**, obtenue soit avec une caméra sur pied, soit avec un appareil photo vidéo.

En revanche, vous pouvez digitaliser en 16 couleurs avec des tons pastels. Voir sur la disquette une image de ce type.

La carte DGA-1

C'est la plus chère, mais comme Lucky Luke, elle digitalise plus vite que son ombre... Vous pouvez donc digitaliser soit en continu, soit image par image.

Deux boutons sur la carte permettent de régler la luminosité et le contraste. Deux prises reliées à la carte permettent de brancher un moniteur de contrôle (moniteur de IIe par exemple) sur l'image qui passe par la carte et la source Vidéo. Moyennant l'achat d'un adaptateur, vous pouvez aussi raccorder la carte sur une télévision couleur. En pratique, vous regardez directement

votre source et le moniteur de contrôle ne s'avère pas indispensable.
Inconvénient de la DGA-1, elle ne digitalise qu'en 16 niveaux de gris, à vous de repasser l'image en couleurs, ce qui est possible mais demande du temps.

Voir sur la disquette, une image digitalisée en noir et blanc et une autre retravaillée en couleurs.

L'équipement vidéo

Attention à ne pas juger la qualité des cartes de digitalisation ci-dessus directement sur l'image GS que vous obtenez. J'ai parlé plus haut de **"chaîne vidéo"** : la carte de digitalisation n'en est qu'un des maillons, et comme dans toute chaîne, **la qualité globale de la chaîne est celle de son moins bon maillon.**

Autrement dit, **meilleure est la source vidéo, meilleure sera l'image GS.** Si votre camescope a un capteur de 290.000 pixels, la qualité sera moins bonne qu'avec un autre camescope équipé d'un capteur de 470.000 pixels (Caméra Sony HI8 CCDV700, 12.000F, par exemple). Votre choix de la carte de digitalisation d'image sera donc orienté par le budget que vous désirez y consacrer, **ainsi que par l'équipement vidéo** dont vous disposez, ou dont vous comptez vous équiper.

Si vous voulez acheter un magnétoscope S-VHS, vous pourrez sans problème sortir le signal vidéo venant du magnétoscope en Pal, et digitaliser vos images avec l'une des cartes Pal ci-dessus, ces magnétoscopes étant équipés pour ressortir, à partir d'une cassette enregistrée en S-VHS, soit en S-VHS, soit en Pal, soit en Sécam.

Comment relier la source vidéo à la carte de digitalisation

Le principe reste le même quel que soit l'appareil de lecture utilisé, que ce soit un magnétoscope, une télévision, un camescope, un vidéodisque ou un appareil photo vidéo : il faut relier la sortie vidéo de l'appareil de lecture sur l'entrée vidéo de la carte de digitalisation.

La sortie de l'appareil peut être une prise Péritel, dans ce cas là vous pouvez trouver un cordon Péritel universel, équipé d'une prise Péritel d'un côté et des câbles vidéo et sons de l'autre. La sortie peut être une prise RCA (Sony), Mini Din (JVC) ou BNC (anciens magnétoscopes), à vous d'acheter (entre 50 et 200 F) ou de fabriquer (entre 6 et 30 F) le câble qui convient.

Chapitre 2 : enregistrer les images GS sur magnétoscope

C'est certainement l'activité vidéo la moins chère pour ceux qui ont déjà un magnétoscope. En attendant les joies de la digitalisation d'images, ou en complément de la digitalisation, pourquoi ne pas enregistrer sur un magnétoscope ce qu'affiche le moniteur de votre IIGS?

Pour quoi faire ?

Du cinéma par exemple, ou bien de l'éducatif, ou bien encore une démonstration de votre appareil.

Du cinéma : avec **VCR Companion** de Broderbund (30$ chez Program Plus), préparez le script de vos titrages de début, de plans de coupe ou de fin, puis lors du montage de vos films, insérez vos animations.
Si vous avez une carte de digitalisation, digitalisez les images de début, plan de coupe et fin, prenez votre programme de dessin et par dessus les images, dessinez vos titres et autres intertitres... avant de les intercaler dans votre montage de films.

Du matériel éducatif : vous en avez assez de répéter tous les jours comment utiliser le traitement de texte? Enregistrez au fur et à mesure les images-écrans des différentes phases du traitement de texte, tout en les commentant au micro, ou bien en post-sonorisant le film.

Encore une fois, je fais confiance à votre imagination pour explorer tous les domaines possibles...

Le matériel

A nouveau deux possibilités s'offrent à vous pour sortir l'image de votre ordinateur sur un magnétoscope. Soit relier directement les deux appareils, ce qui permet d'obtenir une image en noir et blanc, soit intercaler un transcodeur RVB/Sécam, ce qui permet d'obtenir une image en couleurs.

Attention, dans tous les cas, ne pas oublier de mettre le GS en **50 HZ** pour éviter un défilement sur l'écran vidéo.

En noir et blanc : juste un câble

Si l'on désire uniquement une image en **noir et**

blanc, il n'y a pas besoin d'interface spécifique. Le câble suivant suffit (GS DB 15 à gauche, Péritel à droite) :

Pins 1-6-13	..... Pins	4-5-9-13-17-18-21	masse
Pin 2	..... Pin	15	rouge
Pin 12	..... Pin	20	composite
Pin 5	..... Pin	11	vert
Pin 9	..... Pin	7	bleu
Pin 11	..... Pins	2-6	son
Pin 8	..... Pin	8	+12 v

Relier la masse au blindage, bien sûr. La différence avec le câble couleur est Pin 12 vers Pin 20. A priori, ça devrait aussi fonctionner sans les rouge/vert/bleu/+12v, mais j'avoue n'avoir pas essayé.

En couleurs : câble et interface.

Pour sortir en couleurs l'image GS sur un magnétoscope VHS Sécam, il faut un câble et un boîtier d'interface RVB. **Le câble** d'abord :

Il s'agit du câble "GS/Péritel", qui devrait encore être au catalogue Apple. Voici son schéma (GS DB15 à gauche, Péritel à droite, et relier la masse au blindage, bien sûr) :

Pins 1-6-13	..... Pins	4-5-9-13-17-18-21	masse
Pin 2	..... Pin	15	rouge
Pin 3	..... Pin	20	composite
Pin 5	..... Pin	11	vert
Pin 8	..... Pin	8	+12 v
Pin 9	..... Pin	7	bleu
Pin 11	..... Pins	2-6	son

[NDLR : la Documentation Apple dont nous disposons préconise d'intercaler une résistance de 190 Ohms entre le Pin 3 du GS et le Pin 20 de la Péritel, et de relier le Pin 8 de la Péritel au Pin 16 de cette même Péritel, en intercalant une résistance de 360 Ohms.]

L'interface ensuite : pour transformer le signal RVB de l'ordinateur en signal Vidéo Composite Sécam accepté par les magnétoscopes, il suffit de raccorder la prise Péritel du câble ci-dessus à une **Interface CGV PHS 60 Modèle 2S** (disponible par exemple chez Général Vidéo , 700 F environ). Il ne restera plus alors qu'à relier cette Interface à l'entrée Vidéo/Son du magnétoscope (Péritel ou autre), avec un câble *ad hoc*, acheté ou fabriqué suivant la notice fournie avec l'Interface CGV.

Chapitre 3 : la Video Overlay Card et l'incrustation d'images.

Nous consacrerons en fait deux chapitres de cet article à la carte Video Overlay Card d'Apple : elle le mérite bien.

Qu'est-ce que l'incrustation d'images ?

Il faut avant tout distinguer clairement la **digitalisation** des images (chapitre 1) et l'**incrustation** vidéo. Ce ne sont pas les mêmes fonctions, et les cartes qui font l'une ne font pas l'autre. Une carte de **digitalisation** d'images **récupère un fichier image utilisable par votre ordinateur.**

La carte d'**incrustation** d'images **écrit ce qui est affiché sur l'écran de votre ordinateur par dessus une image vidéo**. Cette image vidéo et l'incrustation sont ensuite récupérées sur un magnétoscope. Il ne vous reste plus qu'à concurrencer Roger Rabbit ou Mary Poppins...

Dans tout ce qui suit, nous allons donc bien séparer la notion d'**image vidéo**, qui désigne ce qui vient d'une source vidéo, comme le film de votre camescope par exemple, et celle d'**image ordinateur**, qui désigne toutes les images affichées par les programmes de votre ordinateur.

Ce qui crée la confusion est le fait que **la carte d'incrustation permet de faire apparaître une image vidéo sur l'écran du GS**. Elle reste cependant une image vidéo, destinée à ressortir vers une autre destination vidéo (magnétoscope, télévision etc...), où elle sera montrée et sauvegardée.

L'image vidéo ne fait que transiter par le IIGS, il est donc **impossible de sauvegarder sur disquette l'image vidéo**, avec ou sans incrustation de l'image ordinateur : on retombe à ce moment-là dans le domaine des cartes de digitalisation d'images.

Video Overlay Card : comment la trouver ?

Pour le matériel destiné à l'incrustation, pas d'hésitation possible : il n'existe qu'une seule carte, la **Video Overlay Card** d'Apple. Mais, me direz vous, **Apple France ne la commercialise pas**, elle n'existe qu'aux USA en NTSC !

Eh bien pas du tout, **il existe une version Pal de cette carte**, et elle est commercialisée par Apple : par Apple... Australie ! Bon d'accord, l'Australie est loin et même un bond de kangourou ne suffit pas pour y aller : alors mieux vaut s'adresser en Suisse, chez Musicomp. Pour un prix en Francs Suisses de 1490 F (environ 6000 F français) plus port, elle est à vous.

Si vous programmez, vous pouvez acquérir

pour environ 150 F (suisses) le kit logiciel pour gérer directement l'outil 33 utilisé par la carte, sans passer par l'accessoire VideoMix (ce kit est disponible aussi chez APDA aux USA).

Installation de la Video Overlay Card

La carte s'installe impérativement en slot 3 du GS. (Il faudra tirer sur le câble de votre Transwarp pour installer celle-ci en slot 4, si vous en avez une). Deux connecteurs se raccordent à l'arrière du IIGS. Le premier pour l'entrée et la sortie vidéo (prises RCA), le deuxième pour le moniteur du IIGS, moniteur qui restera à demeure branché sur ce connecteur, même quand la Video Overlay Card ne sera pas utilisée. Aucun réglage à effectuer sur la carte, celle-ci est réglée en usine une fois pour toutes.

Pour les branchements en entrée :

Reliez la source vidéo Pal (camescope par exemple) sur l'entrée de la carte. Si vous rentrez en Sécam, l'image est traitée en noir et blanc, mais si vous utilisez un transcodeur Pal/Sécam en sortie, vous récupérerez l'image vidéo en couleurs à la sortie. Attention par contre à la qualité de l'incrustation, qui sera moins bonne que pour un signal Pal en entrée.

Pour les branchements en sortie :

• Vous avez un magnétoscope et une télévision Pal/Sécam? Aucun problème, enregistrez votre film avec incrustation en Pal.
• Vous avez un magnétoscope seulement Sécam, ou bien une télévision seulement Sécam? Intercalez entre la carte et le magnétoscope un **transcodeur Pal/Sécam** (environ 1000 F chez tous les marchands de camescope ou chez Général Vidéo).

Fonctionnement de la Video Overlay Card

Après avoir parcouru la notice, très claire et émaillée de nombreuses photos, pour éviter tout problème de mauvaise installation, on se lance. On boote le IIGS, au tableau de bord on le configure en 50HZ, et avec une bordure et un écran noir.
On met le camescope en lecture : miracle, l'image vidéo du camescope apparaît sur le moniteur du IIGS avec la pomme qui va et vient par dessus. On lève la tête et sur l'écran de télévision, on voit la même image. !
Allez on se lance un peu plus... On boote GS/OS dans lequel on aura copié le NDA Videomix (soi-même ou avec l'Installer) ainsi que le Tool033, nda et tool livrés avec la carte évidemment.
Une fois sous le Finder, l'image vidéo apparaît sous la bordure (noire) ainsi que sous tous les endroits qui sont de couleur noire. On peut commencer à incruster.

La carte Overlay gère les 16 couleurs à incruster suivant la méthode de 1 plus 15. C'est à dire qu'à un moment donné vous avez choisi une couleur clef, les 15 autres couleurs étant alors non clef. Par exemple, dans les réglages d'origine, la couleur clef est le noir, et l'ensemble des autres couleurs est non clef. Le réglage de l'ensemble des couleurs non clef sera le même pour les 15 couleurs non clef.

Vous pouvez rendre la couleur clef transparente, non transparente ou plus ou moins transparente, même chose pour **l'ensemble** des 15 couleurs non clef, qui peut aussi être transparent, non transparent ou plus ou moins transparent. Le NDA Vidéomix sert à gérer cette transparence.

Vidéomix affiche la palette standard des 16 couleurs. Avec la souris, vous sélectionnez la couleur clef : si une image vidéo transite par votre IIGS, vous la verrez apparaître **sous la couleur clef si celle-ci est transparente**. Vous sélectionnez de deux façons différentes : soit sur la palette de Vidéomix, soit directement sur l'image ordinateur de votre IIGS, dans le cas où cette image n'utilise pas une palette standard.

Vidéomix sert non seulement à sélectionner la couleur clef, mais sert aussi à déterminer son degré de transparence. Une barre de défilement gère la transparence de la couleur clef, une autre barre de défilement gère la transparence de l'ensemble des couleurs non clef.
Pour la couleur clef, vous pouvez la rendre :
• *totalement transparente :* cette couleur sera invisible par-dessus l'image vidéo,
• *pas du tout transparente :* cette couleur apparaîtra parfaitement par-dessus l'image vidéo.
• *en partie transparente :* cette couleur apparaîtra plus ou moins distinctement par dessus l'image vidéo, suivant le réglage de votre barre de défilement.
Vous réglez de la même manière la transparence de l'ensemble des couleurs non clef. Ces possibilités de réglage permettent la superposition à l'image vidéo d'images ordinateur invisibles, fantomatiques ou très nettes, à votre choix.

Autre possibilité offerte par Vidéomix, *le réglage de la couleur ou de la saturation des couleurs de*

l'image vidéo. Une barre de défilement vous permet d'ajouter ou de retirer du vert ou du rouge à votre image vidéo, une autre barre de défilement vous permet de passer votre image vidéo du noir et blanc aux couleurs les plus criardes.

☞ *Hélas, si vous travaillez en Pal, cette modification des couleurs apparaît bien sur l'écran du IIGS, mais à la sortie, sur l'écran de télévision ou en enregistrement sur le magnétoscope, les images vidéo gardent leurs couleurs d'origine (VideoMix, ayant été écrit pour le NTSC, ne gère pas parfaitement le Pal).*

Cela dit, **la qualité de l'image produite par la Video Overlay Card est excellente** : entre une image vidéo qui transite par votre ordinateur, et la même image vidéo copiée directement, on ne voit pas la différence.

Mais si, c'est facile !

Si ce qui précède vous donne l'impression que "tout ça est bien compliqué", rassurez vous : il est toujours difficile d'expliquer avec des mots des choses évidentes pour qui les **voit**. A l'usage, c'est facile, surtout si vous utilisez un logiciel comme **VCR Companion** qui permet de gérer les titres avec défilements horizontaux ou verticaux, les effets spéciaux, les attentes clavier etc.

Vous désirez **refaire Roger Rabbit ?** Utilisez **Cartooners** ou bien **Art and Film Director**. Une fois votre animation ordinateur prête, incrustez-la sur votre image vidéo.

Vous désirez **faire du sous-titrage ?** Utilisez un programme de dessin et réglez enfin chez vous ce que la télévision ne fait jamais, à savoir placer votre texte dans un rectangle en bas de l'écran, et éviter que le sous-titrage ne soit indiscernable, à cause de la couleur de l'image vidéo.

Chapitre 4 : une nouvelle tablette graphique.

Quelle nouvelle tablette graphique ? **La Video Overlay Card, bien sûr !**

Pour comprendre cette idée, il suffit de comparer l'*image vidéo* à un *dessin sur papier* et l'*image ordinateur* à une *feuille de papier calque*. Placez votre calque sur votre dessin et recopiez-le par transparence. De même, avec la carte Overlay, placez votre image vidéo sur le moniteur et avec n'importe quel programme de dessin, recopiez-la pour en faire une *image-ordinateur* !

☞ **Petit mode d'emploi pour décalquer** un dessin, un bâtiment, un portrait etc :

• placez votre camescope sur un pied, ou utilisez un film avec un plan fixe assez long, ou encore utilisez un appareil photo vidéo.
• lancez votre programme de dessin.
• visez (ou lisez) l'image que vous voulez reproduire.

Grâce à la carte Overlay, l'image vidéo prise par la caméra apparaît sur le moniteur de votre GS. Votre programme de dessin met à votre disposition tous les outils pour dessiner, crayon, droite, rectangle, couleurs etc : à vous d'utiliser ces outils pour recopier l'image vidéo et obtenir une image ordinateur, que vous pourrez sauvegarder sur disquette et retravailler ensuite à loisir.

(*Voir sur la disquette un dessin des chevaliers du zodiak, recopié avec cette technique par un utilisateur de GS... âgé de 8 ans*).

L'avantage de la carte par rapport à une tablette graphique, c'est qu'il n'y a pas de limite de taille, ou de lieu : il suffit d'avoir le film ou la photo de l'objet à recopier, alors que la tablette graphique exige absolument une feuille de papier comme modèle.

J'espère avoir répondu à la plus grande partie de vos questions concernant tous les problèmes de cartes de digitalisation, de sortie sur un magnétoscope ou de carte Overlay. N'oubliez pas que le sujet peut paraître compliqué en le lisant, mais qu'**à l'usage, c'est assez simple**.

Adresses :

Convertisseur PAL/SECAM/NTSC : Universal Electronique, 23-29 rue de Stephenson, 75018 Paris.
Visionary GS : Virtual Realities Inc, 4129 University Avenue, Riverside, CA 92501, USA.
Computer Eyes en NTSC : Program Plus, 75 Research Drive, Stratford, CT 06497, USA.
Computer Eyes en Pal : ToolBox, 6 rue Henri Barbusse, 95100 Argenteuil.
DGA-1 en Pal : Microsphère, 118 Boulevard Richard Lenoir, 75011 Paris.
Général Vidéo : 10 Boulevard de Strasbourg, Paris.
CGV : Compagnie Générale de Vidéotechnique, Strasbourg.
Musicomp : Bottmingerstrasse 1, CH 4102 Binningen, Suisse.

[NDLR: ToolBox Mag et ToolBox déclinent toute responsabilité quant à l'exactitude des informations techniques et commerciales données dans cet article, qui n'ont aucune valeur contractuelle. Nous ne sommes pas assez compétents pour vérifier!
Toute précision doit être demandée à l'auteur lui-même, par courrier, sous enveloppe à son nom adressée à ToolBox Mag, avec enveloppe timbrée pour la réponse. Merci!]

Made in France

Bernard Fournier

Crackpoty Boot Selector v1.1

Un utilitaire bien pratique pour choisir le périphérique de boot. Ce petit programme s'installe à la place du fichier de démarrage PRODOS et permet de choisir le lecteur de démarrage en appuyant sur la touche ♁ lors du boot. Une page graphique avec un Welcome sonore du plus bel effet vous permet alors de cliquer sur une des icônes des périphériques en ligne. On peut booter sur une partition du disque dur, sur un lecteur 5 1/4 en slot 6 ou sur un lecteur 3 1/2 en slot 5 (lecteur 1 ou 2 !). On peut même booter directement sur la ROM ou sur la RAM.
Autre fonctionnalité du CBS, on a la possibilité de diminuer automatiquement le volume sonore en fonction de l'heure (les noctambules fanatiques des jeux d'arcade apprécieront).
Un système GS/OS 4.0 modifié est fourni sur la disquette, autorisant le lancement d'applications gourmandes en mémoire lorsqu'on ne dispose que d'un méga de RAM : on peut par exemple installer ce système sur une partition du disque dur et booter directement dessus.
Un shareware modique (50F) qui se révèle vite indispensable.

C & M Band, 25 rue des Alleux, 21490 Saint Julien

TransProg 2.1

Notre éminent collaborateur François Uhrich vient de terminer la dernière version de TransProg. Cet utilitaire permet de lancer une application à tout moment (sans revenir dans le Finder), de lancer une application lors du boot et même de protéger votre GS en demandant un mot de passe lors du lancement du système !
Cette version est considérablement améliorée par rapport à la 1.0 : TransProg est désormais accessible dans la barre des menus (clic sur une icône) ou à n'importe quel endroit de l'écran (♁-Clic). Les applications 'mémorisées' ne sont plus limitées à 40, il fonctionne sur les ROM 01 et 03.
Quelques fonctionnalités annexes trahissent le souci de perfection de l'auteur : purge de la mémoire, réglage de la vitesse du processeur selon l'application lancée, exécution d'une application en cliquant sur un document...
Pour les 90F du shareware, vous aurez en prime **FonteDAInstallateur 2.0** qui permet de lancer ou ôter des accessoires et de charger les fontes de votre choix à partir de toute application.

F. Uhrich, 67 Allée de la Robertsau, 67000 Strasbourg.

MémoCartes 1.0

La délicieuse passionaria du GS nous offre ce mois-ci un autre logiciel pour nos chères têtes blondes... Le principe du jeu est simple : des cartes sont affichées, face cachée; le joueur doit trouver les paires ayant la même figurine. Un excellent jeu faisant appel à la mémoire visuelle et qui convient parfaitement aux enfants de 2 à 99 ans. Il est possible de dessiner ses propres cartes afin de personnaliser ce jeu.
Ce logiciel est diffusé en shareware (100F). Jouez-y **sérieusement** avec votre fils de 7 ans, il est probable qu'il vous flanquera une rouste mémorable : ils sont moins forts en assembleur, mais **ils ont plus de mémoire que nous !**

Carole Fox, Logement de fonction Ecole Azéma, 1 rue Mgr Mondon, 97400 Saint Denis, LA REUNION

Pour figurer dans "Made in France"

Il faut :

- Nous envoyer votre programme complet, avec vos nom et adresse pour nos lecteurs.
- Nous préciser si ce programme est en freeware ou en shareware, et dans ce second cas mentionner le prix complet (port, etc).
- Enfin, ce n'est pas une condition, mais quand les auteurs de shareware, qui font quelque argent avec cette chronique, songent à s'abonner à ToolBox Mag, cela nous encourage à continuer...

Soundsmith 0.95b

Logiciel de composition musicale écrit par l'auteur de *Jigsaw* et *Laser Force*. Ce shareware à 20$ permet également d'inclure vos musiques dans vos propres programmes grâce à l'outil 219 paru dans ToolBox Mag 2. Les sons produits par la carte Sonic Blaster sont récupérables par le logiciel qui utilise par ailleurs des instruments au format ASIF ou PSWV. Entre autres possibilités, vous avez accès à 14 pistes sonores en stéréophonie, vous pouvez définir des effets spéciaux et composer vos musiques en entrant directement les notes.
La version 1.0 (qui ne saurait tarder) sera complètement compatible MIDI et permettra donc de connecter le GS avec un synthétiseur afin de créer aisément des musiques et bibliothèques de sons.
Si vous joignez une boîte de 10 disques vierges à votre chèque bancaire, l'auteur se fera un plaisir de vous fournir des musiques prêtes à être écoutées ainsi que des instruments aussi variés que nombreux (une bonne centaine au bas mot).

Huibert Aalbers, c/Madre Antonia Paris, 6, K2, 12c, 28015 Madrid, ESPAGNE

QSO et QSL de F6GQK

Voici comment F6GQK lui-même présente DxFile :

« Ce soft n'est pas destiné à enregistrer des milliers de Qso, mais plutôt à faire le bilan des Qso réalisés dans le but d'obtenir un diplôme comme le *DXCC*, le *WAZ* ou même le *DDFM*, par exemple. Il vous permet, entre autres possibilités :

- La saisie, la modification et la suppression d'un Qso.
- L'impression de la liste de tous les Qso du fichier.
- L'impression d'une liste triée (fenêtre courante).
- La recherche d'un ou plusieurs Qso, par préfixe, par date ou par pays Dxcc.
- L'impression d'étiquettes pour cartes Qsl. »

Limpide, non ? Pour ceux qui ne visent pas encore le *DDFM*, sachez que ce programme est une gestion de fichier dédiée pour radio-amateurs, qui marche parfaitement. Les radio-amateurs sont des gens fort sympathiques, qui n'ont attendu ni les Modem ni le 3615 pour communiquer tranquilles.

Vous aurez DxFile pour la somme ridicule de 50 F de shareware : il suffit d'avoir un Qso avec F6GQK. Sinon, vous pouvez toujours écrire à l'un de nos lecteurs :

Christian RAMADE, 7 Rue de Quelisoy, 56260 Larmor-Plage

La démo de Noël

Après *Modulae*, voici un nouveau cadeau du FTA: ***Christmas Demo***, une géniale démo graphique/son, avec du texte écrit dans la bordure, etc, en freeware bien sûr.
Une des raisons de ces superbes démos, c'est la confrontation avec quelques programmeurs américains. Eh bien, amis californiens, soyez-en conscients: pour les Ordinateurs et les Pommes, vous êtes les meilleurs, OK.
Mais pour les vins et les animations GS, les crus du Vieux Continent sont insurpassables!

FTA, 34 Rue des Rotondes, 21000 Dijon.

Mac Eraser (♂)

Vous trouverez sur la disquette, dans le catalogue /Mac.Eraser, un petit programme (ERASE.SYSTEM) d'Olivier Goguel. Il ***transforme toute disquette Apple II GS introduite en slot 5, drive 1 de votre GS en disquette 800k Macintosh*** (ce sont les mêmes lecteurs, avec le même formatage physique de disquettes).
Apparemment, le Mac ne connait pas la différence entre "effacer" une disquette (Erase) et "formater" une disquette: il reformate physiquement à tous les coups, au lieu de simplement réécrire un catalogue vierge. Pour transformer une disquette Mac en disquette GS, il suffit d'utiliser la commande "Erase" de GS/OS (avec Prosel par exemple). Pour effacer une disquette Mac, c'est désormais tout aussi simple: il suffit d'un Apple II GS...

Le source de ce petit programme paraîtra dans un prochain ToolBox Mag, car il contient une intéressante "bidouille": la revectorisation des appels SmartPort pour lui faire écrire des blocs de 524 octets (incluant les "tags" du Mac) au lieu de 512.

Notons que ce programme contient un amusant "proverbe du jour". ToolBox Mag, étant un magazine dédié à l'Apple II GS, ne peut se prononcer sur le Mac, et laisse à Olivier la paternité et la responsabilité de son affirmation: *nous ne l'approuvons pas, mais nous ne le désapprouvons pas non plus...*

JYB

Un exemple de programme en TML Pascal II :

MasterMind GS ( )

Raoul Barbieux

Comment jouer à MasterMind GS ( )

Lancez l'application MasterMind GS. Un menu d'aide vous rappellera comment on joue au MasterMind, si vous en avez besoin.

Telle est la force du GS, et des applications "Desktop" produites par TML Pascal II: en fait, aucune explication n'est nécessaire pour s'en servir...

L'outil TML Pascal 2 ( )

La ToolBox du Système 5, un vrai Paradis, mais au bout d'un chemin d'Enfer !!

Pourtant, rouler en TML Pascal 2, c'est le haut de gamme, et avec toutes les options : un éditeur qui n'a rien à envier aux meilleurs traitements de texte, un compilateur à détecteur d'erreur de syntaxe, et surtout une gestion de ressources, la boîte automatique !

Et malgré cela, au début, on patine... Pourtant, il y a de la documentation : le livre de *TML Pascal 2* (en anglais...), l'ouvrage *Boîte à Outils de l'Apple II GS* (en français, mais les exemples sont traités en C...), *Apple II GS Toolbox Reference* (3 énormes volumes, en anglais...). Mais quand on vient du Basic AppleSoft et de l'Assembleur du 6502, tout ça, c'est encore du chinois.

Heureusement, **il existe un outil miracle** : une disquette d'exemples en TML Pascal 2, *"TML Source Code Library 2"*, qui illustre largement les possibilités de la ToolBox. C'est bien fait, c'est indispensable.

Le programme Master Mind que je vous propose est un exemple de plus qui, je l'espère, vous fera gagner du temps, comme ceux de la disquette TML m'en ont fait gagner.

La gestion de Ressources de TML Pascal 2 ( )

La ToolBox est un ensemble de routines que l'on peut appeler en leur "passant" une collection de paramètres en tous genres. Avec la version précédente de TML Pascal ou avec GS Basic, un appel à la ToolBox était une opération lourde, compliquée et parfois décourageante. TML Pascal 2 résume cette opération à renseigner une ou plusieurs fenêtres de dialogue (parfois même renseignées par défaut !) et à la référencer à une Ressource. Cette référence de Ressource est ensuite utilisée comme paramètre dans un appel simplifié à la ToolBox. **Si, si, vraiment, c'est formidable !**

Le Programme MasterMind GS ( )

Le jeu est construit autour d'une image (*"ImageJeu"*) qui représente l'ensemble des décors, c'est à dire la planche de jeu, la boîte à pions, l'affiche, les pions et les clous. Cette image, dessinée avec PaintWorks Gold, c'est le fichier *"MM.JEU"*, (format Paint, compacté). Elle est chargée au démarrage de chaque partie (Procédure *"DemarreJeu"*) par la Procédure *"ChargeImageJeu"*, sous la forme d'une pixelmap représentée par le LocInfo *"ImJeuLI"*; le pointeur du grafport associé est *"gImageJeu"*.
C'est dans ce LocInfo que l'on prélève les décors pour les afficher dans les fenêtres visibles à l'écran pendant la partie, fenêtres définies par les pointeurs de leur grafport *"gFenTrous"*, *"gFenCoul"* et *"gFenRep"*. L'affichage de ces décors s'effectue par l'appel des Procédures *"DessTrous"*, *"DessCoul"* et *"DessRep"*. Ces Procédures sont particulières: dans le listing, elles sont précédées de *"{$DefProc}"*; de plus, dans la

Procédure *"SetUpWindow"*, le troisème paramètre de la déclaration des fenêtres de pointeurs *"gFenTrous"*, *"gFenCoul"* et *"gFenRep"*, est l'adresse de cette procédure particulière, *"@ + nom de la procédure"*. Ces particularités permettent la mise à jour automatique des fenêtres, soit lorsqu'elles sont sélectionnées, soit lorsqu'une fenêtre superposée est retirée, soit lors d'un défilement par des ascenseurs.
Tout au long de la partie, sur la planche de jeu, le joueur va provoquer l'affichage de pions de couleur et la réponse du GS va provoquer l'affichage de clous noirs ou blancs. Ces pions et ces clous sont des portions de l' *"ImageJeu"* et leur affichage, par les procédures *"DessPion"* et *"DessClou"*, se fait d'abord à l'écran dans le grafport de pointeur *"gFenTrous"* de la fenêtre qui affiche la planche de jeu, mais aussi dans le grafport du LocInfo *"ImJeuLI"* pour assurer les mises à jour éventuelles.
Pour une nouvelle partie, il faudra recharger l'*"ImageJeu"* d'origine, puisque la précédente est surchargée par les dessins de pions et de clous.

Les autres fichiers du programme

"MM.DESK.Scr" est le dessin en page de garde dessiné aussi avec PaintWorks Gold (format écran). Il est installé comme fond du bureau dans la procédure *"DessAccueil"*, c'est à dire que c'est le dessin qui sera vu à l'écran si aucune fenêtre n'est sélectionnée. Il a été créé avec la même palette de couleurs que le dessin *"MM.JEU"*, et c'est en le chargeant par la procédure *"ChargeImageDesk"* que l'on installe cette palette à la place de la palette par défaut du Système.
"MM.AIDE" est le fichier Ascii, créé avec AppleWorks GS, qui contient la règle du jeu appelée par le menu Aide.
"MM.PALM" est un fichier binaire qui contient le palmarès des meilleurs scores des joueurs. Un tableau initial a été créé par la procédure *"CreTableauPalm"* où tous les noms sont "INCONNU", toutes les dates sont "10-89" et tous les scores sont de 9. Les scores inférieurs ou égaux à 9 entrent dans le classement par l'algorithme à la fin de la procédure *"GagnePerdu"*.
"MM.TACO" est un fichier de 24 tables de couleurs construit par la procédure *"CreTableCouleur"*. Ces tables sont utilisées par les procédures *"Apparaitre"* et *"Disparaitre"* qui créent les effets de fondu aux changements d'écrans. Charger ce fichier étant plus rapide que l'exécution de la routine qui crée les 24 tables, on adopte !

Quelques tuyaux complémentaires

Barre de Menu et Menus en couleurs : tout simplement par l'instruction *SetBarColors* (procédure *"SetUpMenus"*).

Encadrement de fenêtres en couleurs : un pointeur de paramètres couleur (*"CoulFen"*) a été mis en commun pour toutes les fenêtres (procédure *"SetUpWindows"*).

Agir dans plusieurs fenêtres en gardant toujours la même active : c'est possible ! (procédure *"MainEventLoop"*).

Le curseur en montre, c'est gratuit : il suffit de le saisir pendant qu'il est à l'écran ! (corps principal du programme, en fin de listing).

Des listings imprimés en condensé, c'est de l'économie de papier; voici une suggestion de paramètres d'impression : *" Esc Z Ctrl-G Ctrl-à Esc q Esc B Esc L 0 1 6 "*, qui donnent les caractères américains, compressés, 8 lignes par pouce, une marge de 16 caractères, et 90 lignes par page.

Help ! Help !

A propos de listings, certains caractères dans le listing peuvent se comporter comme des caractères de contrôle pour l'imprimante (exemple : é ou è dans un commentaire entre { }, qui déclenche l'impression des caractères en double largeur à la suite de ce commentaire). Si un lecteur bien inspiré à trouvé **quel caractère peut déclencher un saut de page**, je suis preneur de l'information !

[NDLR: je n'ai pas vérifié, mais je parie un ZX81, une carte 2+ pour Mac LC, ou un Mac 128, que j'ai trouvé ! A tous les coups, il s'agit du "bug du bit 7" ou "bug du AND #$7F" que j'avais signalé dans Pom's. Avant d'imprimer, TML met le bit 7 à zéro. Du coup, les caractères entre $80 et $9F sont envoyés comme caractères de contrôle à l'imprimante. Voyez le tableau pages 18/19, et enlevez $80: "é" et "è" deviennent $0E et $0F, soit Control-N et Control-O ! Comme vous voulez un saut de page (Control-L), il vous faut un $0C + $80 = $8C, c'est-à-dire un "â" (aring). Le reste page 18. Comme quoi, parmi les "has-been du 8 bits", il y a même TML ! Il faut ***vraiment*** *jeter Appleworks-8 ! JYB]*

Comment entrer à une position aléatoire dans la table des nombres aléatoires ? Dans la procédure *"Initialisations"* l'argument de *"SetRandSeed"* a été fabriqué à partir de valeurs des date et heure du système; ça marche, mais il y a sûrement mieux ! Un lecteur a-t-il la solution ?

```
{---------- ♣♣♣ ----------}
Program MasterMind { Raoul Barbieux - version 1 - nov 90 }
{-------------------------------------------------------------}
USES Types, { initialisation des outils }
  Locator, Memory, MiscTool, QuickDraw, { Outil 1-2-3-4 }
  Events, Desk, IntMath, TextTool, { Outil 6-5-11-12 }
  Controls, Windows, Menus, { Outil 16-14-15 }
  LineEdit, Dialogs, QDAux, Print, { Outil 20-21-18-19 }
  Scrap, StdFile, Fonts, Lists, { Outil 22-23-27-28 }
  GSOS, TMLUtils, Resources,
  TextEdit; { Outil 34 }
{-----------}
CONST
  Noir=0; GrisF=4; GrisCl=11; Blanc=15;
  Bleu=1; Marron=2; Vert=3; Rouge=12; Orange=13; Jaune=14;
  { Menus }
    kAppleMenuID=1;
      kAproposItem=301;
    kMenuMenuID=2;
      kNouveauJeuItem=304; kSolutionItem=306; kPalmaresItem=307;
      kAideItem=309; kQuitterItem=311;
  { Fenetres }
    kNomJoueur=22; kForce1=24; kForce2=25;
    kForce3=26; kEsc=27; kOK=28;
    kTexteAide=30;
    kBouton=41;
    kSolPer=45; kStop=46; kOKSol=47;
{-----------}
VAR
  mmid: Integer; gStartStopRef: Ref;
  gMainEvent: EventRecord; Code: Integer; Termine, Sortie: Boolean;
  Montre, Fleche: CursorPtr;
  gFenAide, gFenTrous, gFenCoul, gFenRep,
  gFenNJ, gFenPalm, gFenScore, gFenSol: WindowPtr;
  CoulFen: WindColor;
  I, J, K, L, C: Integer;
  Trou: array[1..5] of record
      RectT, RectF, RectPft, RectPij, RectRft, RectRij: Rect;
  CoulP: Integer; end;
  ChoixC: array[1..9] of record
      RectC,RectP: Rect; Coul: Integer; end;
  Cache: array[1..5] of record
      RectP: Rect; CoulP: Integer; end;
  Clic: Point;
  Horloge: packed array[0..20] of byte; Date: Str255;
  Sec, Min, Heure, Jour, Mois, An: Integer;
  NomJoueur: String[8]; Force, Score: Integer;
  Gagne, UtilSauvePalm: Boolean;
  hCtlNomJoueur: CtlRecHndl; hLENomJoueur: LERecHndl;
  Palm: array[1..3,1..8] of record
      Nom: String[8]; Mois, An, Score: Integer; end;
  CreateRec: CreateRecGS; parmsOpen: OpenRecGS;
  parmsEOF: EOFRecGS; Erreur: Boolean;
  parmsReadWrite: IORecGS; parmsClose: RefNumRecGS;
  DestroyRec: NameRecGS;
  glmScrHndl, glmCompHndl, glmDecompHndl, gTextHndl: Handle;
  PPixParam: PaintParam; MasqHndl: RgnHandle;
  SourceRect,DestRect: Rect;
  DeskLI, ImJeuLI: LocInfo; glmageJeu: GrafportPtr;
  SrcRectD, SrcRectT, SrcRectC, SrcRectR: Rect;
  RectFlecheFT, RectFlecheIJ: Rect;
  TableCouleur: array[0..24] of ColorTable;
{-------------------------------------------------------------}
procedure CreateFichier(NomFichier: Str255);
var
  Fichier: GSString255;
begin
  P2GSString(NomFichier,Fichier);
  with CreateRec do begin
    pCount:= 5;pathname:= @Fichier; access:= $C3;
    fileType:= $06; auxType:= $00; storageType:= $01;
  end;
  CreateGS(CreateRec);
end;
{-----------}
procedure OpenFichierGetEOF(NomFichier: Str255);
var
  Fichier: GSString255;
begin
  P2GSString(NomFichier,Fichier);
  with parmsOpen do begin
    pCount:= 2; pathname:= @Fichier;
  end;
  OpenGS(parmsOpen);
  if _ToolErr <> noError then begin
    Erreur:= true;
    Exit(OpenFichierGetEOF);
  end;
  with parmsEOF do begin
    pCount:= 2; refnum:= parmsOpen.refnum;
  end;
  GetEOFGS(parmsEOF);
end;
{-----------}
procedure ParmsReadWriteFichier(DestPtr: Ptr; FileSize: LongInt);
begin
  with parmsReadWrite do begin
    pCount:= 4; refnum:= parmsOpen.refnum;
    databuffer:= DestPtr; requestCount:= FileSize;
  end;
end;
{-----------}
procedure CloseFichier;
begin
  with parmsClose do begin
    pCount:= 1; refnum:= parmsOpen.refnum;
  end;
  CloseGS(parmsClose);
end;
{-------------------------------------------------------------}
procedure ChargeImageDesk;
var
  srcTablePtr: Ptr;
begin
  OpenFichierGetEOF('MM.DESK.Scr');
  glmScrHndl:= NewHandle(parmsEOF.eof,mmid,
  attrPage+attrFixed+attrLocked,nil);
  ParmsReadWriteFichier(glmScrHndl^, parmsEOF.eof);
  ReadGS(ParmsReadWrite);
  CloseFichier;
  srcTablePtr:= Pointer(Ord4(glmScrHndl^)+$7E00);
  BlockMove(srcTablePtr,Pointer($E19E00),512);
  with DeskLI do begin
    portSCB:= GetMasterSCB;
    ptrToPixImage:= glmScrHndl^;
    width:= 160;
    SetRect(boundsRect,0,0,320,200);
  end;
  HUnlock(glmScrHndl);
end;
{-----------}
procedure ChargeImageJeu;
var
  DstPtr: Ptr; DstSize: Integer; Ignore: Integer;
begin
  OpenFichierGetEOF('MM.JEU');
  glmCompHndl:= NewHandle(parmsEOF.eof,mmid,
  attrPage+attrLocked,nil);
  ParmsReadWriteFichier(glmCompHndl^, parmsEOF.eof);
  ReadGS(ParmsReadWrite);
  CloseFichier;
  glmDecompHndl:= NewHandle(2*$7D00,mmid,
  attrPage+attrFixed+attrLocked,nil);
  DstSize:= 2*$7D00; DstPtr:= glmDecompHndl^;
  Ignore:= UnPackBytes(pointer(ord4(glmCompHndl^)+$222),
  parmsEOF.eof-$222,DstPtr,DstSize);
```

```
  with ImJeuLI do begin
    portSCB:= GetMasterSCB;
    ptrToPixImage:= gImDecompHndl^;
    width:= 160;
    SetRect(boundsRect,0,0,320,400);
  end;
  OpenPort(gImageJeu); SetPortLoc(ImJeuLI);
  DisposeHandle(gImCompHndl); HUnlock(gImDecompHndl);
end;
{-----------------------------------------}
procedure ChargeTexteAide;
begin
  OpenFichierGetEOF('MM.AIDE');
  gTextHndl := NewHandle(parmsEOF.eof, mmid,attrLocked, nil);
  ParmsReadWriteFichier(gTextHndl^, parmsEOF.eof);
  ReadGS(ParmsReadWrite);
  CloseFichier;
  TESetText(teTextIsHandle + teDataIsTextBlock,
    TETextRef(gTextHndl),GetHandleSize(gTextHndl),0,TEStyleRef(nil),
    TEHandle(GetCtlHandleFromID(gFenAide, kTexteAide)));
  HUnlock(gTextHndl);
end;
{-----------------------------------------}
procedure SauveTableauPalm;
begin
  OpenFichierGetEOF('MM.PALM');
  ParmsReadWriteFichier(@Palm[1,1].Nom, 3*8*15);
  WriteGS(ParmsReadWrite);
  CloseFichier;
end;
{--------------------}
procedure CreTableauPalm;
begin
  for I:= 1 to 3 do for J:= 1to 8 do begin
    with Palm[I,J] do begin
      Nom:= 'Inconnu'; Mois:= 7; An:= 90; Score:= 9;
    end;
  end;
  CreateFichier('MM.PALM'); SauveTableauPalm;
end;
{--------------------}
procedure ChargeTableauPalm;
begin
  OpenFichierGetEOF('MM.PALM');
  if Erreur then begin
    CreTableauPalm; Erreur:= false;
  end
  else begin
    ParmsReadWriteFichier(@Palm[1,1].Nom, 3*8*15);
    ReadGS(ParmsReadWrite);
    CloseFichier;
  end;
end;
{-----------------------------------------}
procedure SauveTableCouleur;
begin
  OpenFichierGetEOF('MM.TACO');
  ParmsReadWriteFichier(@TableCouleur[0], 25*16*2);
  WriteGS(ParmsReadWrite);
  CloseFichier;
end;
{--------------------}
procedure CreTableCouleur;
var
  Coul: Integer;
begin
  GetColorTable(0,TableCouleur[0]); SetColorTable(7,TableCouleur[0]);
  for I:= 1 to 8 do begin
    for J:= 0 to 15 do begin
      Coul:= GetColorEntry(7,J); K:= Coul;
      if K> $7FF then Coul:= Coul-$100;
      if K< $700 then Coul:= Coul+$100;
      SetColorEntry(8,J,Coul); K:= K-$100*trunc(K/$100);
      if K> $7F then Coul:= Coul-$10; if K< $70 then Coul:= Coul+$10;
      SetColorEntry(9,J,Coul); K:= K-$10*trunc(K/$10);
      if K> $7 then Coul:= Coul-$1; if K< $7 then Coul:= Coul+$1;
      SetColorEntry(10,J,Coul);
    end;
    for J:= 0 to 2 do GetColorTable(8+J,TableCouleur[3*I-2+J]);
    SetColorTable(7,TableCouleur[3*I]);
  end;
  CreateFichier('MM.TACO'); SauveTableCouleur;
end;
{--------------------}
procedure ChargeTableCouleur;
begin
  OpenFichierGetEOF('MM.TACO');
  if Erreur then begin
    CreTableCouleur; Erreur:= false;
  end
  else begin
    ParmsReadWriteFichier(@TableCouleur[0], 25*16*2);
    ReadGS(ParmsReadWrite);
    CloseFichier;
  end;
end;
{-----------------------------------------------------------------}
procedure Disparaitre;
begin
  for I:= 1 to 24 do for J:= 1 to 40 do SetColorTable(7,TableCouleur[I]);
end;
{--------------------}
procedure Apparaitre;
begin
  for I:= 23 downto 0 do for J:= 1 to 40 do
SetColorTable(7,TableCouleur[I]);
end;
{-----------------------------------------}
function RectDec(Rectangle: Rect): Rect;
begin
  OffSetRect(Rectangle,0,-GetContentOrigin(gFenTrous).v);
  RectDec:= Rectangle;
end;
{-----------------------------------------}
procedure PPixels(FenDestPtr: WindowPtr; Mask: Boolean);
begin
  MasqHndl:= NewRgn; OpenRgn; FrameOval(DestRect);
CloseRgn(MasqHndl);
  with PPixParam do begin
    ptrToDestLocInfo:= LocInfoPtr(FenDestPtr);
    maskHandle:= nil; if Mask then maskHandle:= Handle(MasqHndl);
  end;
  PaintPixels(PPixParam);
  DisposeRgn(MasqHndl);
end;
{-----------------------------------------------------------------}
{$DefProc}
procedure DessAccueil;
begin
  PPToPort(LocInfoPtr(@DeskLI),SrcRectD,0,0,modeCopy);
end;
{--------------------}
{$DefProc}
procedure DessTrous;
begin
  PPToPort(LocInfoPtr(@ImJeuLI),SrcRectT,0,0,modeCopy);
end;
{--------------------}
{$DefProc}
procedure DessCoul;
begin
  PPToPort(LocInfoPtr(@ImJeuLI),SrcRectC,0,0,modeCopy);
end;
{--------------------}
{$DefProc}
procedure DessRep;
```

```
begin
  PPToPort(LocInfoPtr(@ImJeuLI),SrcRectR,0,0,modeCopy);
  MoveTo(12,9); DrawString(Concat('Joueur : ',NomJoueur));
  MoveTo(162,9); DrawString(Concat('Force  ',Int2String(Force)));
  DrawControls(gFenRep);
end;
{----------}
{$DefProc}
procedure DessScore;
begin
  DrawControls(gFenScore);
  SetPort(gFenScore); MoveTo(30,20);
  if Gagne then
  DrawString(Concat('Gagne au ',Int2String(Score),'eme Coup'))
  else DrawString(Concat('Perdu au ',Int2String(Score),'eme Coup'));
end;
{----------}
{$DefProc}
procedure DessAide;
begin
  DrawControls(gFenAide);
end;
{----------}
procedure DessPion;
begin
  SourceRect:= ChoixC[Trou[C].CoulP].RectP;
  DestRect:= RectDec(Trou[C].RectPft); PPixels(gFenTrous, true);
  DestRect:= Trou[C].RectPij; PPixels(@ImJeuLI, true);
end;
{----------}
procedure GlissePion;
var
  N, Dx, Dy: Integer; Dest: Rect;
begin
  if I<> 9 then begin
    Dest:= RectDec(Trou[C].RectPft);
    Dx:= 260-Dest.Left; Dy:= 19*I-40-Dest.Top;
    for N:= trunc((Dx-52)/25) downto 1 do begin
      Dy:= trunc(25*N*Dy/Dx); Dx:= 25*N;
      DestRect:= Dest; OffSetRect(DestRect,Dx,Dy);
      if DestRect.Top>= 0 then begin
        SourceRect:= ChoixC[Trou[C].CoulP].RectP;
        PPixels(gFenTrous, true);
        SourceRect:= Trou[C].RectPij; OffSetRect(SourceRect,Dx,Dy);
        PPixels(gFenTrous, true);
      end;
    end;
  end;
  DessPion;
end;
{----------}
procedure DessClou;
begin
  DestRect:= RectDec(Trou[K].RectRft); PPixels(gFenTrous, true);
  DestRect:= Trou[K].RectRij; PPixels(@ImJeuLI, true);
end;
{----------}
procedure GlisseClou;
var
  N, H: Integer; RectClou: Rect;
begin
  H:= RectDec(Trou[K].RectRft).Top; RectClou:= SourceRect;
  for N:= trunc(H/8) downto 1 do begin
    DestRect:= RectDec(Trou[K].RectRft); OffSetRect(DestRect,0,-8*N);
    PPixels(gFenTrous, true);
    SourceRect:= Trou[K].RectRij; OffSetRect(SourceRect,0,-8*N);
    PPixels(gFenTrous, true);
    SourceRect:= RectClou;
  end;
  DessClou;
end;
{----------}
procedure DescendFleche;
begin
  SetRect(SourceRect,263,239,275,276);
  DestRect:= RectDec(RectFlecheFT); PPixels(gFenTrous, false);
  DestRect:= RectFlecheIJ; PPixels(@ImJeuLI, false);
end;
{----------}
procedure DessSol;
var
  Rectangle: Rect;
begin
  SetRect(Rectangle,12,11,127,30);
  SetPort(gFenRep);
  SetSolidPenPat(GrisCl); PaintRect(Rectangle);
  SetSolidPenPat(Marron); FrameRect(Rectangle);
  OffSetRect(Rectangle,0,13);
  SetPort(gImageJeu);
  SetSolidPenPat(GrisCl); PaintRect(Rectangle);
  SetSolidPenPat(Marron); FrameRect(Rectangle);
  for I:= 1 to 5 do begin
    SourceRect:= ChoixC[Cache[I].CoulP].RectP;
    DestRect:= Cache[I].RectP; PPixels(gFenRep, true);
    OffSetRect(DestRect,0,13); PPixels(@ImJeuLI, true);
  end;
end;
{----------}
procedure GagnePerdu;
var
  Bouton: Str255; hCtlBouton: CtlRecHndl;
begin
  DessSol;
  Score:= L;
  if gagne then Bouton:= 'BRAVO' else Bouton:= 'BEURK';
  hCtlBouton:= GetCtlHandleFromID(gFenScore,kBouton);
  {SetCtlTitle(Bouton,hCtlBouton);}
  SelectWindow(gFenScore); ShowWindow(gFenScore);
  SetCursor(Fleche^);
  gMainEvent.wmTaskMask:= $001FFF77;
  sortie:= false;
  repeat
    code:= TaskMaster($FFFF, gMainEvent);
    case code of
      wInDesk, wInMenuBar, wInContent, wInFrame: SysBeep;
      wInControl: sortie:= true;
    end;
  until sortie;
  Disparaitre;
  HideWindow(gFenScore);
  if Gagne then begin
    if Score<= Palm[Force,8].Score then begin
      UtilSauvePalm:= true;
      for I:= 8 downto 1 do begin
        if Score> Palm[Force,I].Score then leave;
      end;
      if Score<= Palm[Force,1].Score then I:= 0;
      if I< 7 then begin
        for J:= 8 downto I+2 do begin
          Palm[Force,J].Nom:= Palm[Force,J-1].Nom;
          Palm[Force,J].Mois:= Palm[Force,J-1].Mois;
          Palm[Force,J].An:= Palm[Force,J-1].An;
          Palm[Force,J].Score:= Palm[Force,J-1].Score;
        end;
      end;
      Palm[Force,I+1].Nom:= NomJoueur;
      Palm[Force,I+1].Mois:= Mois;
      Palm[Force,I+1].An:= An;
      Palm[Force,I+1].Score:= Score;
    end;
  end;
  HideWindow(gFenRep);
  HideWindow(gFenCoul); HideWindow(gFenTrous);
  ClosePort(gImageJeu); DisposeHandle(gImDecompHndl);
  DisableMItem(306);
  Apparaitre;
```

```
  gMainEvent.wmTaskMask:= $001FFF7F;
end;
{-----------------------------------------}
procedure PositPion;
begin
  SetPort(gFenTrous); GlobalToLocal(Clic);
  for I:= 1 to 5 do begin
    If PtInRect(Clic,RectDec(Trou[I].RectT)) then begin
      DessPion; C:= I; Leave;
    end;
  end;
end;
{-----------------}
procedure ChoixCouleur;
begin
  SetPort(gFenCoul); GlobalToLocal(Clic);
  for I:= 1 to 9 do begin
    If PtInRect(Clic,ChoixC[I].RectC) then begin
      Trou[C].CoulP:= I; GlissePion; inc(C); if C=6 then C:= 5; Leave;
    end;
  end;
end;
{-----------------}
procedure Reponse;
var
  ScrollV: Integer; MaskFlecheHndl: RgnHandle;
  Test: array[1..5] of record
      Prop, Sol: Boolean; end;
begin
  { noircir 'REPONSE' }                    {/////////}
  SetCursor(Montre^);
  DessPion;
  for I:= 1 to 5 do begin
    Test[I].Sol:= true; Test[I].Prop:= true;
  end;
  K:=1;
  for I:= 1 to 5 do begin
    if Trou[I].CoulP= Cache[I].CoulP then begin
      SetRect(SourceRect,299,247,307,254); GlisseClou;
      Test[I].Sol:= false; Test[I].Prop:= false; inc(K);
    end;
  end;
  if K= 6 then begin
    Gagne:= true; GagnePerdu;
  end
  else begin
    for I:= 1 to 5 do begin
      if Test[I].Prop then begin
        for J:= 1 to 5 do begin
          if Test[J].Sol then begin
            if Trou[I].CoulP= Cache[J].CoulP then begin
              SetRect(SourceRect,299,262,307,269); GlisseClou;
              Test[J].Sol:= false; Test[I].Prop:= false; inc(K);
              Leave;
            end;
          end;
        end;
      end;
    end;
    if L= 12 then GagnePerdu
    else begin
      C:= 1; inc(L);
      ScrollV:= GetContentOrigin(gFenTrous).v;
      if ScrollV <= 22*(L-7)-6 then
      SetContentOrigin(0,22*(L-7),gFenTrous);
      DescendFleche;
      OffSetRect(RectFlecheFT,0,22); OffSetRect(RectFlecheIJ,0,22);
      for I:= 1 to 5 do begin
        with Trou[I] do begin
          CoulP:= 9;
          OffSetRect(RectT,0,22); OffSetRect(RectF,0,22);
          OffSetRect(RectPft,0,22); OffSetRect(RectPij,0,22);
          OffSetRect(RectRft,0,22); OffSetRect(RectRij,0,22);
        end;
      end;
    end;
  end;
  SetCursor(Fleche^);
end;
{-----------------------------------------------------------------}
procedure DemarreJeu;
var
  NbreCoul: Integer; Coul: array[1..5] of Integer;
begin
  SetCursor(Montre^);
  NbreCoul:= 8; if Force= 3 then NbreCoul:= 9;
  for I:= 1 to 5 do begin
    J:= abs(random); Coul[I]:= trunc(NbreCoul*(J/32769))+1;
  end;
  if Force= 1 then begin
    for I:= 2 to 5 do for J:= 1 to I-1 do begin
      if Coul[I]= Coul[J] then begin
        inc(Coul[I]);
        if Coul[I]= 9 then Coul[I]:= 1;
        dec(I); leave;
      end;
    end;
  end;
  for I:= 1 to 5 do Cache[I].CoulP:= Coul[I];
  Gagne:= False; L:= 1; C:= 1;
  ChargeImageJeu;
  SetRect(RectFlecheFT,0,4,12,41); SetRect(RectFlecheIJ,0,50,12,87);
  for I:= 1 to 5 do begin
    with Trou[I] do begin
      CoulP:= 9;
      SetRect(RectT,I*22-7,2,I*22+13,19);
      SetRect(RectF,I*22,8,I*22+6,13);
      SetRect(RectPft,I*22-6,3,I*22+12,18);
      SetRect(RectPij,I*22-6,49,I*22+12,64);
      SetRect(RectRft,I*14+141,7,I*14+149,14);
      SetRect(RectRij,I*14+141,53,I*14+149,60);
    end;
  end;
  ShowWindow(gFenRep); ShowWindow(gFenCoul);
  SetContentOrigin(0,0,gFenTrous); ShowWindow(gFenTrous);
  SelectWindow(gFenTrous);
  EnableMItem(306); SetCursor(Fleche^);
  Apparaitre;
end;
{-----------------}
procedure NouveauJeu;
var
  theID: Integer;
begin
  Disparaitre;
  HideWindow(gFenRep); HideWindow(gFenCoul);
HideWindow(gFenTrous);
  ShowWindow(gFenNJ);
  Apparaitre;
  LESetSelect(0,length(NomJoueur),hLENomJoueur);
  gMainEvent.wmTaskMask:= $00130006;
  sortie:= false;
  repeat
    code:= TaskMaster($FFFF, gMainEvent);
    case code of
      wInDesk, wInMenuBar, wInContent, wInDrag, wInFrame: SysBeep;
      wInControl: begin
        theID:= gMainEvent.wmTaskData4;
        case theID of
          kOK: begin
            sortie:= true;
            HandToPtr(LEGetTextHand(hLENomJoueur),
              @NomJoueur[1],LEGetTextLen(hLENomJoueur));
            NomJoueur[0]:= Chr(LEGetTextLen(hLENomJoueur));
            Disparaitre;
            HideWindow(gFenNJ);
```

```
            DemarreJeu;
          end;
          kEsc: begin
            sortie:= true;
            DisableMItem(306);
            HideWindow(gFenNJ);
          end;
          kForce1: Force:= 1;
          kForce2: Force:= 2;
          kForce3: Force:= 3;
        end;
      end;
      updateEvt: begin
        if WindowPtr(gMainEvent.message)= gFenNJ then begin
          BeginUpdate(gFenNJ); DrawControls(gFenNJ);
          EndUpdate(gFenNJ);
        end;
      end;
    end;
  until sortie;
  gMainEvent.wmTaskMask:= $001FFF7F;
end;
{-------------------------------------------------------------------}
procedure Apropos;
var
  Ignore: Integer;
begin
  Ignore:= AlertWindow(4,nil,ref(1));
end;
{-------------------------------------------}
procedure Solution;
var
  theID: Integer;
begin
  SelectWindow(gFenSol); ShowWindow(gFenSol);
DrawControls(gFenSol);
  gMainEvent.wmTaskMask:= $001FFF77;
  sortie:= false;
  repeat
    code:= TaskMaster($FFFF, gMainEvent);
    case code of
      wInDesk, wInMenuBar, wInContent: SysBeep;
      wInControl: begin
        theID:= gMainEvent.wmTaskData4;
        case theID of
          kSolPer: SysBeep;
          kStop: begin
            sortie:= true;
            HideWindow(gFenSol); SelectWindow(gFenTrous);
          end;
          kOKSol: begin
            HideWindow(gFenSol);
            Gagne:= False; GagnePerdu;
          end;
        end;
      end;
    end;
  until sortie;
  gMainEvent.wmTaskMask:= $001FFF7F;
end;
{-------------------------------------------}
procedure Palmares;
begin
  SelectWindow(gFenPalm);
  ShowWindow(gFenPalm); SetPort(gFenPalm);
  SetBackColor(GrisCl);
  MoveTo(0,13);
  DrawString('    Force 1        Force 2        Force 3    ');
  MoveTo(0,21);
  DrawString(' Nom  Dat Pt  Nom  Dat Pt  Nom  Dat Pt ');
  SetBackColor(Jaune);
  for J:= 1 to 8 do for I:= 1to 3 do begin
    if Palm[I,J].Mois< 10 then K:= 1 else K:= 0;
    MoveTo(I*105-103,26+j*10); SetForeColor(Bleu);
    DrawString(Copy(Concat(Palm[I,J].Nom,'      '),1,6+K));
    MoveTo(I*105-52+8*K,26+j*10); SetForeColor(GrisF);
    DrawString
      (Concat(Int2String(Palm[I,J].Mois),'.',Int2String(Palm[I,J].An)));
    MoveTo(I*105-13,26+j*10); SetForeColor(Rouge);
    DrawString(Int2String(Palm[I,J].Score));
    SetForeColor(Noir);
  end;
  gMainEvent.wmTaskMask:= $001FFF77; { sauf menuselect }
  sortie:= false;
  repeat
    Code:= TaskMaster($FFFF,gMainEvent);
    case Code of
      wInDesk,wInMenuBar,wInContent{,wInDrag,wInFrame}: SysBeep;
      wInGoAway: sortie:= true;
    end;
  until sortie;
  HideWindow(gFenPalm); SelectWindow(gFenTrous);
  gMainEvent.wmTaskMask:= $001FFF7F;
end;
{-------------------------------------------}
procedure Aide;
begin
  SelectWindow(gFenAide); ShowWindow(gFenAide);
  gMainEvent.wmTaskMask:= $001FFF77; { sauf menuselect }
  sortie:= false;
  repeat
    Code:= TaskMaster($FFFF,gMainEvent);
    case Code of
      wInDesk, wInMenuBar, wInContent: SysBeep;
      wInGoAway: sortie:= true;
    end;
  until sortie;
  HideWindow(gFenAide);SelectWindow(gFenTrous);
  gMainEvent.wmTaskMask:= $001FFF7F;
end;
{-------------------------------------------}
procedure Quitter;
begin
  Termine:= true;
  Disparaitre;
  if UtilSauvePalm then SauveTableauPalm;
  DisposeAll(mmid);
end;
{-------------------------------------------------------------------------}
procedure MainEventLoop;
var
  MenuChoisi,ItemChoisi: Integer; gFenSelect: WindowPtr;
begin
  gMainEvent.wmTaskMask:= $001FFF7F; { sauf SelectWindow }
  Termine:= false;
  repeat
    if FrontWindow = gFenTrous then begin
      SetPort(gFenTrous);
      InvertOval(RectDec(Trou[C].RectF));
    end;
    for I:= 1 to 30 do begin
      Code:= TaskMaster($FFFF,gMainEvent);
      case Code of
        wInSpecial, wInMenuBar: begin
          MenuChoisi:= HiWrd(gMainEvent.wmTaskData);
          ItemChoisi:= LoWrd(gMainEvent.wmTaskData);
          case ItemChoisi of
            kAproposItem: Apropos;
            kNouveauJeuItem: NouveauJeu;
            kPalmaresItem: Palmares;
            kSolutionItem: Solution;
            kAideItem: Aide;
            kQuitterItem: Quitter;
          end;
          HiliteMenu(false,MenuChoisi);
        end;
```

```
        wInContent: begin
         GetMouse(Clic); LocalToGlobal(Clic);
         gFenSelect:= pointer(gMainEvent.wmTaskData);
         if gFenSelect= gFenTrous then PositPion;
         if gFenSelect= gFenCoul then ChoixCouleur;
         if gFenSelect= gFenRep then Reponse;
        end;
      end;
    end;
  until Termine;
end;
{-----------------------------------------}
procedure SetUpMenus;
var
  Height: Integer;
begin
  SetSysBar(NewMenuBar2(RefIsResource, ref(1), nil));
  SetMenuBar(nil);
  FixAppleMenu(kAppleMenuID); { ajoute les NDA }
  Height:= FixMenuBar;
  SetBarColors(Rouge+GrisCl*16,Orange,Marron*16);
  DrawMenuBar;
end;
{-----------------------------------------}
procedure SetUpVariables;
var
  Caract: Char;
begin
  ReadAsciiTime(@Horloge); Date:= '';
  for I:= 0 to 19 do begin
    Caract:= chr(BAnd(Horloge[I],$7F)); Date:= Concat(Date,Caract);
  end;
  Sec:= String2Int(Copy(Date,16,2)); Min:= String2Int(Copy(Date,13,2));
  Heure:= String2Int(Copy(Date,10,2));
  Jour:= String2Int(Copy(Date,1,2));
  Mois:= String2Int(Copy(Date,4,2)); An:= String2Int(Copy(Date,7,2));
  SetRandSeed(Sec*100+Min*10+Heure); Erreur:= false;
  ChargeTexteAide; ChargeTableauPalm; ChargeTableCouleur;
  SetColorTable(7,TableCouleur[0]); for I:= 13 to 199 do SetSCB(I,7);
  Force:= 1; NomJoueur:= 'Inconnu';
  Score:= 12; UtilSauvePalm:= False;
  hCtlNomJoueur:= GetCtlHandleFromID(gFenNJ,kNomJoueur);
  hLENomJoueur:= LERecHndl(hCtlNomJoueur^^.ctlData);
  LESetText(@NomJoueur[1],length(NomJoueur),hLENomJoueur);
  SetRect(SrcRectT,0,46,225,309);
  SetRect(SrcRectC,241,13,317,198);
  SetRect(SrcRectR,0,13,241,44);
  for I:= 1 to 5 do SetRect(Cache[I].RectP,I*22-6,13,I*22+12,28);
  for I:= 1 to 9 do begin
    SetRect(ChoixC[I].RectC,13,19*I-9,64,19*I+6);
    SetRect(ChoixC[I].RectP,236,17*I+187,254,17*I+202);
  end;
  ChoixC[1].Coul:= Blanc; ChoixC[2].Coul:= Noir;
  ChoixC[3].Coul:= Marron;
  ChoixC[4].Coul:= Bleu; ChoixC[5].Coul:= Vert;
  ChoixC[6].Coul:= Jaune;
  ChoixC[7].Coul:= Orange; ChoixC[8].Coul:= Rouge;
  ChoixC[9].Coul:= GrisF;
  with PPixParam do begin
    ptrToSourceLocInfo:= LocInfoPtr(@ImJeuLI);
    ptrToSourceRect:= @SourceRect;
    ptrToDestPoint:= @DestRect.TopLeft;
    mode:= modeCopy;
  end;
end;
{-----------------------------------------}
procedure SetUpWindows;
begin
  gFenAide:= NewWindow2(nil,0,@DessAide,nil,
      RefIsResource,Ref(1),rWindParam1);
  gFenTrous:= NewWindow2(nil,0,@DessTrous,nil,
      RefIsResource,Ref(2),rWindParam1);
    MoveWindow(1,46,gFenTrous); SizeWindow(225,153,gFenTrous);
    SetPort(gFenTrous); SetSolidBackPat(9);
  gFenCoul:= NewWindow2(nil,0,@DessCoul,nil,
      RefIsResource,Ref(3),rWindParam1);
    MoveWindow(243,14,gFenCoul); SizeWindow(76,186,gFenCoul);
    SetPort(gFenCoul); SetSolidBackPat(GrisF);
  gFenRep:= NewWindow2(nil,0,@DessRep,nil,
      RefIsResource,Ref(4),rWindParam1);
    MoveWindow(1,14,gFenRep); SizeWindow(241,31,gFenRep);
    SetPort(gFenRep); SetSolidBackPat(GrisCl); SetBackColor(GrisCl);
  gFenNJ:= NewWindow2(nil,0,nil,nil,
      RefIsResource,Ref(5),rWindParam1);
  gFenPalm:= NewWindow2(nil,0,nil,nil,
      RefIsResource,Ref(6),rWindParam1);
    SetPort(gFenPalm); SetSolidBackPat(Jaune); SetBackColor(Jaune);
  gFenScore:= NewWindow2(nil,0,@DessScore,nil,
      RefIsResource,Ref(7),rWindParam1);
    SetPort(gFenScore); SetSolidBackPat(Orange);
    SetBackColor(Orange);
  gFenSol:= NewWindow2(nil,0,nil,nil,
      RefIsResource,Ref(8),rWindParam1);
    SetPort(gFenSol); SetSolidBackPat(Vert); SetBackColor(Vert);
  with CoulFen do begin
    FrameColor:= Marron*16;
    TitleColor:= Noir+GrisCl*16+Bleu*256;
    tBarColor:= GrisCl+Orange*16+2*256; {2=lignes}
    GrowColor:= Noir+Vert*16;
    InfoColor:= Jaune*16;
  end;
  SetFrameColor(@CoulFen,nil);
end;
{-----------------------------------------}
procedure Accueil;
var
  ignore: Ptr;
begin
  ChargeImageDesk;
  SetRect(SrcRectD,0,0,320,200);
  ignore:= Desktop(5,Ord4(@DessAccueil));
end;
{-----------------------------------------------------------------}
begin
  mmid:= MMStartUp;
  gStartStopRef:= StartUpTools(mmid,RefIsResource,Ref(1));
  if _ToolErr= noError then begin
    Accueil;
    SetUpWindows;
    SetUpVariables;
    SetUpMenus; DisableMItem(306);
    Montre:= GetCursorAdr; InitCursor; Fleche:= GetCursorAdr;
    MainEventLoop;
  end;
  ShutDownTools(RefIsHandle, gStartStopRef);
end.
{-----------------------------------------------------------------}
```

B S

Il y avait déjà ToolBox, il y a maintenant une deuxième société commerciale française produisant des logiciels pour Apple II GS. Elle s'appelle "BrainStorm SoftWare", et son premier produit est la magnifique version 2.1 du TransProg de... François Uhrich! Tous nos vœux de succès à BS.

Une petite remarque cependant: j'approuve BS d'avoir choisi pour son (très beau) logo un portrait de François Uhrich, vu la valeur des programmes de notre éminent collaborateur. Mais franchement, François, ce n'est pas le portrait de toi qui t'avantage le plus...

JYB

Les ressources du squelette

(Le Resource Manager , deuxième partie - ♂♂)

Bernard Fournier

Dans ToolBox Mag 2, nous avons commencé l'étude des ressources en apprenant à désassembler un fichier ressources à l'aide de l'utilitaire DEREZ fourni avec APW; ou mieux, avec GENESYS. Dans cet article, nous allons voir comment sont constituées les ressources et leurs emplois afin de faciliter la programmation, et enfin comment utiliser au mieux les manuels de référence (en l'occurrence le volume 3 de la ToolBox Reference *[TBRef 3]*, les librairies du langage employé [par exemple *TML//*] et la librairie *Types.Rez*).

Avant de poursuivre plus avant, je rappelle les différences fondamentales entre la création d'un **fichier** de ressources et la création d'un **source** de ressources.
A l'aide de Genesys, vous créez aisément vos ressources: pour cela le mode d'emploi du logiciel est suffisamment explicite. Cependant, le programmeur Apple // GS est souvent atteint d'une maladie incurable: la *"décorticomanie"*. En effet, il ne lui suffit pas de créer ses ressources avec un logiciel, certes très performant, mais il veut savoir "comment" et "pourquoi". Il ne faut pas croire que ce souci relève de la psychiatrie; il s'agit bien au contraire d'une démarche on ne peut plus normale. On ne peut maîtriser pleinement un sujet (ici la programmation) que si on en connaît parfaitement tous les constituants et tous les mécanismes à mettre en oeuvre.
De plus, Genesys ne permet pas de construire un texte *formaté* (c'est à dire un texte comportant des codes de mise en page: style, taille, couleur et format des caractères). On est donc amené à savoir construire ses ressources "à la main": c'est à dire à savoir créer un source compilable avec REZ.
J'insiste beaucoup sur cet aspect de la programmation des ressources, car bien des difficultés proviennent de la méconnaissance de la constitution des ressources.

Dans cet article nous allons analyser certaines ressources: ***StartUpTools, rMenuBar, rMenuItem*** et ***rPString*** qui nous permettront de comprendre la constitution d'une ressource et donc de pouvoir ultérieurement définir les nôtres.

Nous allons pour cela créer ce que l'ami Uhrich appelle un **squelette**: une mini-application, qui ne fait rien du tout, mais qui constituera un cadre dans lequel rentrer vos applications "réelles". J'espère vous convaincre que, grâce aux ressources, vous pouvez vous permettre d'écrire un programme vraiment **squelettique**: en quelques lignes, vous créez une application complète.

Sur la disquette ToolBox Mag 3, vous trouverez le listing complet de ces ressources (fichier AppMini.S) compilable avec REZ: *compile AppMini.S keep=AppMini.R*. Le source de l'application en TML (fichier AppMini.P) est à lier avec le fichier ressources AppMini.R avant compilation.
Consultez également les sources relatifs à PIC.MASTER.GS parus dans ToolBox Mag 2, en particulier LoadSavePic.S qui est le listing exhaustif des ressources employées.

La ressource *StartUpTools*

Toute application DeskTop utilise des appels à la ToolBox. Afin que ces appels se fassent correctement, il faut au préalable désigner les Tools qui seront utilisés. L'initialisation des Tools se fait avec la fonction *StartUpTools:* cette fonction admet trois paramètres en entrée (voir TBRef 3 page 51-18):

```
StartUpTools(
  userID,           { identifiant retourné par
                      le Memory Manager }
  StartStopRefDesc, { type de la référence stockée
                      dans StartStopRef :
        - 0: référence par un Pointeur
        - 1: référence par un Handle
        - 2: référence par un Identifiant ressource }
  StartStopRef);    { référence à la structure
                      contenant la liste des outils }
```

La liste de nos outils étant stockée sous forme de ressource, on va donc avoir un appel comme suit:

```
gStartStopRef:=StartUpTools(MemoryID, 2, Ref(1));
```

ou encore en utilisant des constantes:

```
gStartStopRef:=StartUpTools(MemoryID,
RefIsResource,Ref(kStartStopResID));
```

• *MemoryID*: un entier retourné par la fonction *MMStartUp*
• *RefIsResource*: constante égale à 2 définie dans la librairie *TYPES* de TML
• *kStartStopResID*: constante égale à 1 définie dans les *CONST* en en-tête du source et en corrélation avec l'identifiant de la ressource *rToolStartUp*.
• *gStartStopRef*: référence sur la table des outils initialisés. Cette référence sera employée dans *ShutDownTools* au moment du QUIT. Le type de cette référence est déterminé par le type de *StartStopRefDesc*:

StartStopRefDesc	*gStartStopRef*	Valeur
pointeur	pointeur	0
handle	handle	1
ResID	handle	1

Ayant employé une référence par Identifiant ressource, la référence retournée par *StartUpTools* est donc un handle: dans la fonction *ShutDownTools* on va donc écrire:

```
ShutDownTools(1,gStartStopRef);
```

ou mieux:

```
ShutDownTools(RefIsHandle,gStartStopRef);
```

Dialogue avec les lecteurs

Dans les articles d'initiation au Pascal ou aux ressources, il m'arrive d'employer des termes techniques ou de faire référence à des installations d'utilitaires dans un environnement de programmation. Il peut arriver que certaines de ces "évocations" soient un tantinet ésotériques, ou pour le moins pas évidentes pour le débutant.

Si vous souhaitez avoir des éclaircissements sur une procédure de travail, ou si vous voulez des informations complémentaires sur un sujet, n'hésitez pas à nous écrire: toutes vos demandes seront traitées dans les prochains numéros de ToolBox Mag. Cette revue est le lien entre les programmeurs *de tous niveaux*, et le dialogue est le moyen privilégié de développer l'esprit Apple //.

B.F.

Il faut également définir la ressource *rToolStartup* dans le fichier compilable par REZ. Consultons le TBRef 3 page E-69:

```
resource rToolStartUp (Identifiant, attributs)
  {flag,          // toujours 0
  videoMode,      // mode graphique QuickDraw
  resFileID,      // positionné par StartUpTools
  dPageHandle,    // positionné par StartUpTools
  numTools,       // nombre d'outils employés
  toolArray};     // liste des outils
```

La liste des outils à passer dans ToolArray étant constituée comme suit:

```
{toolNumber1, minVersion1; // n° d'outil et version
                           // de l'outil
[...]
toolNumber2, minVersion2};
```

Note: le n° de version permet de vérifier si l'outil se trouvant sur le système de boot est bien en conformité avec les fonctions employées. En effet, les outils évoluent au fil du temps: des fonctions nouvelles sont implémentées, des corrections sont parfois effectuées... il faut donc être sûr que l'utilisateur dispose des outils que l'on a utilisés pour développer l'application. Si la version de l'outil employé est inférieure au numéro indiqué dans *minVersion*, la fonction *StartUpTools* retourne une erreur; le programmeur doit tester si une erreur est générée par *StartUpTools*, et dans ce cas forcer le programme à quitter (et éventuellement afficher un message signalant l'erreur).
En TML, on fera donc un test sur *_ToolErr* immédiatement après l'initialisation:

```
gStartStopRef:=StartUpTools(...);
if _ToolErr= NoError then
  begin
  .... code de l'application....
  end;
ShutDownTools(....);
```

Revenons à l'examen de la ressource *rToolStartUp*. En examinant la librairie *Types.Rez*, on constate que la ressource *rToolStartUp* est définie comme suit:

```
type rToolStartUp
  {integer=0;     // flag toujours égal à 0
  integer Mode320=0, Mode640=$80;     // mode de
                  // résolution QuickDraw
  integer =0;     // resFileID
  longInt =0;     // dPageHandle
  integer =$$countof(ToolRecs);   // nombre d'outils
  array ToolRecs{
    integer;      // numéro d'outil
    integer;      // version minimum
  }};
```

En examinant cette définition de type, on constate que certains paramètres sont pré-définis (*flags*, *resFileID* et *dPageHandle*) ou calculés

Les Attributs d'une ressource

A chaque ressource est associé un attribut, définissant la manière dont la ressource sera utilisable. Le Resource Manager stocke cette information dans le champ *resAttr* de la structure de la ressource. Il existe trois fonctions permettant d'éditer/modifier cet attribut:

- *GetResourceAttr*
- *SetResourceAttr*
- *MarkResourceChange*

La plupart des attributs définissent la manière dont la ressource sera stockée en mémoire lors de son chargement et sont en liaison directe avec les indicateurs de *NewHandle* du MemoryManager. L'attribut est un entier (deux octets) dont chaque bit renseigne le ResourceManager sur la manière dont la ressource sera gérée. Voici la liste de ces attributs:

- *AttrLocked*	bit	15	0	mémoire non verrouillée
			1	mémoire verrouillée (Move et Purge interdits)
- *AttrFixed*	bit	14	0	mémoire non fixée
			1	mémoire fixée (Move interdit)
- *ResConverter*	bit	11	0	pas de routine de conversion
			1	emploi d'une routine de conversion
- *ResAbsLoad*	bit	10	0	chargement en mémoire libre
			1	chargement à une adresse spécifique
- *AttrPurge3*	bits	9-8	11	purge de niveau 3
- *AttrPurge2*	bits	9-8	10	purge de niveau 2
- *AttrPurge1*	bits	9-8	01	purge de niveau 1
- *AttrPurge0*	bits	9-8	00	purge de niveau 0
- *ResProtected*	bit	7	0	ressource modifiable
			1	ressource non-modifiable
- *ResPreload*	bit	6	0	ressource non chargée lors d'OpenResourceFile
			1	ressource chargée automatiquement lors d'OpenResourceFile
- *ResChanged*	bit	5	0	ressource non modifiée en mémoire
			1	la ressource a été modifiée en mémoire
- *AttrNoCross*	bit	4	0	chevauchement de banc autorisé
			1	chevauchement de banc interdit
- *AttrNoSpec*	bit	3	0	mémoire spéciale utilisable
			1	mémoire spéciale non utilisable
- *AttrPage*	bit	2	0	pas d'alignement mémoire
			1	alignement mémoire requis

Lorsqu'on définit une ressource, on a vu que sa définition se faisait sur le modèle:

resource resType(resID, attribut)

Afin de spécifier l'attribut, on a deux possibilités: soit on indique la valeur (ex $C018), soit on indique les mnémoniques:

resource resType(resID, $C018)

ou alors

resource resType(ResId, attrLocked, attrFixed, attrNoCross, attrNoSpec)

Par suite d'une lacune, ces mnémoniques ne sont pas disponibles dans *Types.Rez;* il convient donc de les définir soi-même (employer alors *Types.Rez.Aux* fourni sur la disquette ToolBox Mag 2 avec une directive du type *#include "préfixe d'accès à Types.Rez.Aux"*).

(*numTools*); et que seuls certains autres sont à définir par l'utilisateur (*videoMode, ToolNumber* et *minVersion*). Ceci signifie que dans notre source de ressource, on n'aura qu'à passer les paramètres utilisateurs en respectant leur format (ici ce sont tous des entiers).
Ceci donne donc:

```
resource rToolStartUp(Tool_Sartup_Table, $0)
    // voir encadré pour la signification
    // de l'attribut $0
{mode640,
{3, $300,
{...}
34, $101
}}
```

et on définit en tête du source la constante *Tool_Startup_Table*:

```
#define Tool_Startup_Table $1
```

D'une manière générale, un fichier source de ressources est constitué comme suit (les // en début de ligne permettent d'inclure des commentaires dans un source REZ):

```
        // exemple standard de fichier REZ
#include "préfixe d'accès à la librairie Types.Rez"
        // liste des définitions des identifiants,
        // classés par Types de ressources
#define ResIdentifiant1  Constante1
[...]
#define ResIdentifiantN  ConstanteN
        // liste des définitions des Types
resource ResType(ResIdentifiant, attributs)
  {
  {liste des paramètres};
  };
```

Chacune des ressources système (c'est à dire les ressources définies par Apple) est commentée et détaillée dans le chapitre E du TBRef 3. Lister également la librairie *Types.Rez* ainsi que son

De la lecture pour les longues soirées d'hiver

Le programmeur Apple // GS doit impérativement posséder certains ouvrages de base et quelques logiciels. En ce qui concerne les logiciels, le choix se fera en fonction du langage de programmation utilisé:

- Assembleur: APW, ORCA ou Merlin
- Pascal: ORCA ou Complete Pascal (anciennement TML Pascal //)
- C: APW ou ORCA

[ByteWorks, éditeur de ORCA, propose des cours de programmation en C et Pascal sur des classeurs avec disquette d'accompagnement comportant les sources commentés étudiés dans les leçons.]

Outre le manuel de référence du langage employé, il est indispensable de se procurer:

- les 3 tomes d'*Apple // GS ToolBox Reference*(Addison Wesley)
- un livre d'approfondissement dans le langage employé:
  - ASM: *Programming the Apple // GS in C and Assembly Language*(H.W. Sams & Co)
  - Pascal: *Topiques Pascal*(par John Colibri, Menemodyne)
  - C: *Le langage C*(par Kernighan et Ritchie, Masson)

Pour ceux qui veulent aller plus loin, quelques ouvrages spécifiques:

- *Exploring Apple GS/OS and ProDos 8*(par Little, Addison Wesley)
- les deux tomes du *GS/OS Reference*(APDA)
- *Apple // GS Firmware Reference*(Addison Wesley)
- *Apple Numerics, 2nd Edition*(Addison Wesley)
- *Le //GS épluché*(ToolBox™)

Tous ces ouvrages, à l'exception du *GS épluché* et de *GS/OS Reference*, sont disponibles en France chez Infothèque, 32 rue de Moscou, 75008 PARIS. Les ouvrages de référence du GS et le manuel "//GS épluché" sont disponibles chez ToolBox. *GS/OS Reference* est disponible chez APDA.

Et bien entendu, imprimez les librairies du langage employé. Pour chaque langage, ces librairies se trouvent dans les dossiers suivants:

- *Pascal TML:* librairies dans le dossier */ToolIntfSrcs*, les noms sont du type *xxx.P*.
- *ORCA Pascal:* librairies dans le dossier */Tool.Interfaces*sur la disquette *Samples*.
- *APW/C ou ORCA/C:* dans le dossier */C.Libraries/CInclude*, les noms sont du type *xxx.H*.
- *l'assembleur APW ou ORCA:* dans dossier */A.Libraries/AInclude*, les noms sont du type *E16.xxx*.
- *l'assembleur Merlin:* dans le dossier */Tool.Equates*, les noms sont du type *E16.xxx.S*
- *les ressources:* dans le dossier */R.Libraries*,lister *Types.Rez* et son complément *Types.Rez.Aux* fourni sur la disquette ToolBox Mag 2.

Muni de ces 50 kg de lecture, le programmeur débutant va passer quelques soirées à lire ses nouveaux livres de chevet. Mais ces achats sont des investissements sûrs, une garantie de trouver tout ce dont il a besoin. Seuls les *Topiques Pascal*, *Le Langage C* et bien sûr le *GS épluché* étant en français, la connaissance de l'anglais est recommandée. Mais un "Basic English" rudimentaire suffit largement.

complément *Types.Rez.Aux* (sur la disquette ToolBox Mag 2) afin de connaître les constantes employées et les paramètres pré-définis pour chaque type de ressource.

La ressource *rMenuBar*

Autre exemple de constitution de ressource, la *rMenuBar* qui va permettre d'élaborer la barre de menus. Cette ressource va nous montrer comment **référencer des ressources dans une ressource.**

rMenuBar: type $8008 (lire TBRef3 page E-55)

définition de la ressource $8008:

```
resource rMenuBar(Identifiant, attributs)
{{version,          // toujours à 0
menubar flag,       // type des références aux menus
menuRefArray};      // liste des menus associés à la barre
```

Consultons le fichier Types.Rez:

```
type rMenuBar
  {integer =0;      // toujours à 0
  integer = $8000;  // type des références aux menus
  array {longInt;}; // table des références aux menus
  longInt =0;};     // terminaison: toujours 0
```

Que constate-t-on? Les seuls paramètres à indiquer seront les références aux menus; ces références étant des identifiants de ressource ($8000 = 100 000 000 000 000 binaire; les bits 15-14 étant à 1-0, les références se font par identifiant de ressource comme indiqué dans TBRef3 page E-55).

Concrètement, on définira notre ressource *rMenuBar* de la manière suivante:

```
#define App_Menu_Bar $1
#define Apple_Menu $1
#define File_Menu $2
#define Edit_Menu $3

resource rMenuBar(App_MenuBar, $C018)  // voir
      // encadré pour la signification des
      // attributs $C018
  {Apple_Menu,
  File_Menu,
  Edit_Menu,
  };};
```

Cette ressource fait appel à d'autres identifiants de ressources (les menus), il convient donc de les définir:

Les ressources dans les ressources

Ressource *rMenu:*

rMenu: type $8009 (lire TBRef3 page E-52)

Définition de la ressource $8009:

```
resource rMenu(Identifiant, attributs)
{{version,      // toujours à 0
menuID,         // n° d'identifiant du menu
menuFlag,       // type d'affichage du menu
menuTitleRef,   // référence au nom du
                // menu (une chaîne)
itemRefArray}; // liste des items associés au menu
```

Consultons le fichier Types.Rez:

```
type rMenu
  {integer = 0;   // toujours 0
  integer;        // identifiant du menu
  integer;        // type du menu
  longint;        // référence à la chaîne de titre du menu
  array {longint;};  // liste des items
  longint = 0;};  // terminaison: toujours 0
```

Avec cette ressource, tous les paramètres (à l'exception de la Version et de la Terminaison) sont à définir par l'utilisateur:

exemple avec le menu File:

```
resource rMenu(File_Menu, $C018)  // voir encadré
                  // pour la signification des
                  // attributs $C018
  {$2;            // identifiant du menu FILE
  $A008;          // type du menu
  File_pString;   // référence à la chaîne titre
    {Open_Menu_Item,  // liste des items
    Close_MenuItem,
    Quit_Menu_Item};};
```

Le type du menu ($A008 ou 1 010 000 000 001 000) signifie:

- bits 15-14 à 1-0: référence sur le titre par un identifiant de ressource
- bits 13-12 à 1-0: références sur les items par identifiants à des ressources
- bits 11-9: toujours à 0
- bit 8 à 0: pas d'appel à un CustomMenu
- bit 7 à 0: menu sélectionnable
- bit 6 à 0: toujours à 0
- bit 5 à 0: inversion lors de la sélection sans utiliser XOR
- bit 4 à 0: menu standard
- bit 3 à 1: menuCache permis (un menu déroulé une fois est "photographié"en mémoire ce qui permet de l'afficher instantanément la fois suivante)
- bits 2-0: toujours à 0

Ressource *rPstring*

Le titre de ce menu est défini par un identifiant à une ressource de type *rPString*.

rPString: type $8006 (page E-59 du TBRef3)

définition de la ressource $8006:

```
rPString (identifiant, attributs)
  {lengthByte;          // octet de longueur de la chaîne
  stringCharacters;};   // les octets de la chaîne
```

consultons Types.Rez:

```
type rPString {pstring;} // chaîne pascal
```

ce qui nous donne:

```
resourcerPString (File_pString, $C018)   // voir encadré
                                         // pour l'attribut
  {" Fichier "};
```

Ressource *rMenuItem*

En ce qui concerne les Items, on va utiliser des ressources *rMenuItem*:

rMenuItem: type $800A (TBRef3 page E56)

définition dans Types.Rez:

```
type rMenuItem
  { integer = 0;   // toujours 0
    integer;       // identifiant de l'item
    char;          // équivalent clavier
    char;          // autre équivalent clavier
    integer;       // caractère à afficher pour les items cochés
    integer;       // type de l'item
    longint;};     // référence à la chaîne du nom de l'item
```

Les paramètres à passer sont aisément compréhensibles depuis que nous avons analysé les ressources rMenu et rPString, aussi je ne m'étendrai pas plus sur le sujet (voir bien entendu TBRef3 page E-57 pour plus amples informations).

Exemple avec l'item OPEN:

```
#define Open_Menu_Item $102
#define Open_String $102
resource rPString(Open_String, $C018)
  {"Ouvrir"};
resource rMenuItem(Open_Menu_Item, $C018)
  {$102,
  "O","o",
  nil,
  $8040,
  Open_String};
```

Remarque: l'identifiant du *MenuItem* et de la chaîne associée ont été tous deux mis à $102. En fait, on peu très bien utiliser des valeurs différentes: le *MenuItem* référence un identifiant Chaîne de valeur quelconque. Mais dans un souci d'homogénéité, il est plus confortable d'avoir des identifiants semblables.

J'espère que cet article vous aura permis de mieux comprendre comment sont constituées les ressources. Au vu de leurs mécanismes, vous en saisirez mieux l'importance dans la programmation. Dans le prochain numéro de ToolBox Mag, nous verrons deux points: la création de ses propres types de ressources et la modification en mémoire ou sur disque d'une ressource.

SoundTrack Tool, suite (☆☆)

Il y avait quelques bugs dans le Tool219, merci à Anthony Cassaigne de les avoir signalés. Voici donc la ***version 1.1 du SoundTrack Tool***, à jour pour SoundSmith 0.95b. Remplacez donc, dans votre catalogue /SYSTEM/TOOLS, la version 1.0 par celle du catalogue /TOOL219.V.1.1.

A noter: une version différente du Tool219 est également présente sur la disquette de SoundSmith 0.95b. A vous de voir quelle version vous préférez utiliser. Mais attention: dans les deux cas, si vous voulez diffuser cet outil dans vos productions, réglez d'abord les questions de droits, soit avec ToolBox Mag, soit avec Huibert Aalbers (*SoundSmith n'est pas dans le domaine public*).

Anthony Cassaigne et Yvan Kœnig nous ont écrits les interfaces pour divers outils de programmation. Dans le sous-catalogue /TOOL219.V.1.1, vous trouverez donc les fichiers suivants:

/TOOL219.V.1.1/SOURCE.MERLIN/TOOL219.S: c'est le source Merlin du Tool219, par Olivier Goguel.

/TOOL219.V.1.1/HELP.MERLIN/SOUNDTRACK: fichier de Help pour Merlin, par Anthony Cassaigne.

/TOOL219.V.1.1/MACROS.APW/M16.SOUNDTRACK: fichier de macros pour l'Assembleur APW/Orca, par Anthony Cassaigne.

/TOOL219.V.1.1/MACROS.MERLIN/TOOLBOXMACS.S: fichier de macros pour Merlin, par Yvan Kœnig. A noter que ce fichier contient les deux types de macros Merlin (macros normales et "super-macros"), et qu'il inclut aussi les macros pour l'outil "Juke-Box" de Stephan Hadinger paru dans notre numéro 1.

/TOOL219.V.1.1/NIFTY.LIST/NLIST.DATA2: texte à couper-coller, selon les instructions données au début du fichier, dans le fichier NLIST.DATA de Nifty List, par Anthony Cassaigne.

/TOOL219.V.1.1/ ORCA.DISASM/DISASM.DATA2: texte à couper-coller, selon les instructions données au début du fichier, dans le fichier DISASM.DATA d'Orca Disassembler, par Anthony Cassaigne.

Merci à tous !

Babar avec nous !

Encore un joyeux drille qui rejoint le gang: c'est en fanfare que nous accueillons ce polisson de ***Jean-Pierre Charpentier*** au Conseil de Rédaction de ToolBox Mag. Etant donné le nombre de claviers qu'il a chez lui, il n'aura pas énormément de temps pour ToolBox Mag. Mais n'oublie pas, Babar, que tu nous a promis une étude sur le Tool de SynthLab "dès que tu pourrais". JYB

GS News

Eric Weyland (♂)

Ce qu'il faut savoir

Le 22 à Asnières

Curieux de connaître le délai d'acheminement par les PTT d'un numéro de ToolBox Mag, je me suis abonné. Expérience intéressante: j'habite à 20 minutes de ToolBox; ma revue postée le 4 décembre d'Argenteuil a mis 15 jours pour trouver ma boîte à lettres (reçue le 20 décembre). Certains abonnés lointains (Ile de la Réunion) ont, semble t-il, eu plus de chance. Nous étudions la possibilité de poster depuis là-bas...

Incider / A+: changement de politique

Le magazine américain dans son numéro de décembre 1990 fait sa couverture sur le Macintosh LC. D'après l'éditorial de Dan Muse, le Rédacteur en chef, la revue s'oriente maintenant sur les ordinateurs de la gamme Apple II et Macintosh; l'Apple II continuant à être le principal centre d'intérêt d'Incider / A+. Il précise à ses lecteurs que la revue s'efforcera de montrer comment l'Apple II et le Macintosh peuvent travailler et communiquer ensemble.

Cette ligne correspond parfaitement aux idées d'Apple en la matière (voir dans le numéro précédent les extraits de la lettre ouverte de John Sculley). L'Apple II et le Mac sont effectivement des machines complémentaires, particulièrement grâce au réseau AppleTalk.

Si vous voulez savoir par exemple sur quelle machine est réalisé ToolBox Mag, eh bien il est réalisé ***sur un réseau AppleTalk***. La pièce principale de ce réseau est une LaserWriter Apple, on y trouve également, en périphérique, divers ordinateurs Apple (GS et Mac), quelques ImageWriters, des lecteurs SCSI Syquest, des lecteurs 800k, etc.

A part quelques confusions *("cette cartouche-là, c'est pour le Mac, pour le GS, ou pour la Laser?")* et quelques conflits *("qui c'est qui a encore renommé la Laser pour se la bloquer pour lui tout seul?")*, tout ça tourne suffisamment rond pour faire votre magazine préféré...

De plus en plus de programmes sont communs Mac/GS: le Système, HyperCard...; les périphériques sont les mêmes: imprimantes, disques durs, lecteurs de disquettes, de cartouches, de CD-Rom, scanners, etc. Au prix où sont aujourd'hui les Mac Plus d'occasion, voilà des périphériques bon marché du GS.

Complete Technology: Mises à jour

TML Systems ayant vendu les droits de ses produits Apple II (entre autre TML Basic et TML Pascal, voir ToolBox Mag 2) à Complete Technology, les utilisateurs des produits TML se demandaient s'ils continueraient à recevoir les mises à jour.

La réponse est oui, mais pour bénéficier des offres d'upgrades ou d'updates, il faut payer à Complete Technology un "droit d'entrée": $19.95 pour TML Basic, $29.95 pour TML Pascal. Pour les détails pratiques, voir Complete Technology.

System 5.03: compléments

Une des améliorations "visibles" se situe dans le Standard File que les applications utilisent pour sélectionner les fichiers; au niveau du choix du volume, au lieu d'utiliser la touche Tab ou le bouton DISK pour passer en revue tous les volumes en ligne, il y a maintenant un bouton VOLUMES qui affiche la liste de tous les volumes en ligne. Il suffit simplement d'ouvrir le volume qui vous intéresse, et d'y utiliser ce que vous y recherchez. Cela est très pratique pour les utilisateurs ayant des volumes à accès plutôt lent (CD Rom ou serveur AppleShare).

Notons aussi que le Standard File fonctionne maintenant "à la Mac", mais de façon encore plus intelligente: si vous introduisez une disquette pendant que le Standard File est actif, non seulement il la reconnaît, mais encore il se place automatiquement sur le catalogue principal de cette disquette. Regardez cela de près avec le Traceur de ce numéro...

En revanche, le 5.03 ne contient pas le driver annoncé pour les imprimantes HP; constatant que les drivers de Vitesse (Harmonie) et de Seven Hills (Independance) étaient meilleurs que le sien, Apple a volontairement renoncé au sien, pour éviter que son driver "gratuit" ne freine la vente de produits commerciaux externes meilleurs. Comme quoi, s'il est (comme tous les autres) capable du pire, Apple (et pas les autres) reste capable du meilleur: **bravo Apple pour ta sportivité.**

Système 5.03: Comment se le procurer?

Très facile: achetez Platinum Paint! Vous aurez alors une version limitée du système. Tout le monde n'a pas la chance d'être développeur et de recevoir à domicile un compact disque contenant le système 5.03 et HyperCard GS. En fait, ce système ne sera pas commercialisé (il a été retiré de la vente); il vous faudra attendre le système 5.04 (HyperCard GS devrait être livré avec le 5.04). Pour la francisation de ce nouveau système, et d'HyperCard, consulter Apple France...

Laser Computers: nouvelle machine?

La société américaine travaille sur un compatible Apple portable. La compagnie cherche un partenaire financier pour lancer la production.

Laser Computers voudrait commercialiser un équivalent de son compatible PC 4 MS-DOS: cet ordinateur portable intègre dans ses roms un système d'exploitation et un logiciel d'application.

Skeleton Desktop Application (Sandy Mossberg)

A l'origine publié par la revue Call Apple, Skeleton Desktop Application est maintenant disponible sur disquette pour les assembleurs APW et ORCA/M ainsi que pour les compilateurs APW C et ORCA/C. Elle contient tous les articles écrits par Mossberg (format AppleWorks), y compris ceux qui devait sortir dans Call Apple. Le prix de la disquette est de $20.

Duet (Cirtech)

La société écossaise a décidé de reporter le projet Duet à une date ultérieure. La mise sur le marché des nouveaux Macintosh par Apple, qui a empêché la campagne publicitaire sur le GS

promise par Apple pour Décembre, a incité Cirtech à différer la fabrication de la carte Duet, qui techniquement est prête. Apple n'ayant pas une politique commerciale très claire vis à vis de l'Apple IIGS, la production de la Duet est un trop gros risque.
Si le Mac LC persiste à avoir du mal à s'imposer sur le marché de l'éducation américaine, il est fort probable que Duet reverra le jour avant les calendes grecques...

Pénurie de jeux?

Sur tous les micro-ordinateurs, on assiste depuis quelques mois à une raréfaction du nombre de programmes de jeu. Les utilisateurs deviendraient ils sérieux? La balle verte d'Arkanoid II ne ferait elle plus recette?
Pas du tout! Le marché du logiciel de jeu se porte de mieux en mieux, mais pas sur micro-ordinateur. Peu à peu, les consoles de jeu sont en train de dévorer, dans le domaine ludique, les micro-ordinateurs.
Dans ce domaine, Nintendo domine de très loin le marché mondial. Sur ce marché, il y a actuellement cinq constructeurs: Nintendo (80% du marché mondial), Sega, Nec, Atari et Amstrad.
Leur stratégie est simple: fabriquer des consoles de jeu en masse pour les vendre à vil prix , et vendre des cartouches de jeu à un prix confortable en n'oubliant pas qu'il n'existe que peu de phénomène de piratage sur les consoles (essayez donc de pirater sans clavier et sans disquettes!).
Ceci explique pourquoi Epyx, Cinemaware et Taïto ont arrêté leurs activités dans l'édition de jeux sur micro-ordinateurs. John Brooks, le programmeur de Rastan, développe maintenant pour les consoles Nintendo...
Cela pose une question de fond: ***l'ordinateur personnel multi-fonctions, comme l'Apple II GS, a-t-il encore de l'avenir, face aux machines dédiées à une application, ou à un type d'applications?***
Notre réponse, à ToolBox Mag, serait en gros celle-ci: il y a une place, et elle est en train de se révéler, pour les fast-foods. Mais il y aura toujours un bon paquet de gens (nous en sommes) qui aimeront la bonne cuisine: ceux-là continueront à aller au Restaurant, et à se faire la cuisine eux-mêmes, au moins en partie...

Le Hard

Zip Chip GS: Précisions

Il existe trois modèles de Zip Chip GS:
- le modèle 1500 (Zip Chip GS): vitesse 8 MHz avec 8 Ko de mémoire cache. Ce modèle n'est pas compatible DMA (Direct Memory Access).
- le modèle 1525 (Zip Chip GSX): vitesse 8 MHz avec 16 Ko de mémoire cache. Ce modèle est compatible DMA.
- le modèle 1600 (Zip Chip GSX PLUS): vitesse 8 MHz avec 16 Ko de mémoire cache. Ce modèle est compatible DMA. D'après Zip Technologies, un kit sera bientôt disponible pour upgrader le modèle 1600 à 12 MHz.

(Applied) Ingenuity: support technique

Après la fermeture de la société américaine, de nombreux utilisateurs de l'Inner Drive se demandaient à qui s'adresser en cas de coup dur.
Le nom du sauveur est Bill Heineman de Custom Software. Cette société californienne commercialise depuis peu une carte d'extension mémoire et un modèle SCSI de l'Inner Drive.

Le soft: les jeux

Dragon Wars (InterPlay)

Jeu de rôle, Dragon Wars donne la possibilité de reprendre les personnages créés dans Bard's Tale 1, 2 ou 3. Comme toujours, la première chose à faire est de créer une vaillante équipe, en veillant à ce que les caractéristiques de chacun des personnages soient complémentaires.
C'est certainement le meilleur jeu de rôle sur Apple IIGS après Dungeon Master, au point de vue des graphiques et des sons, mais aussi du point de vue de l'originalité du scénario. Au tout début du jeu, vous vous trouvez au purgatoire, d'où nul ne peut s'échapper... A vous de prouver le contraire!

Le soft: les utilitaires

HyperCard GS (Apple Computer)

Non! Vous avez bien lu, ce n'est pas une farce! Ce n'est pas un poisson d'avril! Il existe, il marche à la perfection (attention: système 5.03 et 1.5 Mo de ram obligatoire): HyperCard GS version 1.0 B21.
HyperCard est ce que l'on appelle un produit hypermédia: cela signifie que l'on peut facilement créer, en deux coups de cuiller à pot, des programmes ayant de l'allure. Il est facile de créer des cartes contenant des graphiques, d'y insérer du texte, d'y placer des boutons de navigation qui créent des liens entre les différentes cartes; on peut compliquer le tout en ajoutant des sons digitalisés ou des séquences vidéo.
Le langage de programmation HyperTalk permet pas mal de fantaisies... Il permet de créer des scripts; c'est un langage à part entière, proche du Pascal, qui donne un contrôle total sur la pile et les objets qu'elle renferme.
Sur Apple IIGS, il existait jusqu'à aujourd'hui HyperStudio et Nexus, que l'on considérait comme des programmes hypermédia. Ils sont maintenant dépassés!
Au fait, la version Apple IIGS d'HyperCard a une grande supériorité sur celle du Macintosh. Vous ne devinez pas? Elle est en couleurs! La version 2.0 d'HyperCard Mac ne sait toujours pas vraiment ce que c'est.
Au passage, j'en profite pour dire qu'HyperCard Mac n'est plus donné que sous forme de "Runtime" avec chaque Macintosh. La version complète d'HyperCard est un produit commercial ordinaire, diffusé moyennant finances par Claris aux USA, et par Apple en France.
Comment se débarrasser de la concurrence? Facile! Il suffit de donner pendant des années HyperCard pour bloquer le développement d'autres produits hypermédia pour, par la suite, une fois la disparition des concurrents constatée, vendre son produit sans encombres. Quand je pense qu'on a osé parfois, aux USA, se plaindre du "dumping" des Japonais!
Pour la version française d'HyperCard GS, comme pour sa commercialisation en France, à vous de voir auprès d'Apple France s'il en est capable, et pour quand.
En pratique, l'utilisation d'HyperCard GS est identique à celle d'HyperCard Mac. Les deux logiciels sont des frères jumeaux! Il y a là, nul ne pourra

le nier, un formidable travail de développement de la part d'Apple USA. C'est dans un sens rassurant pour l'avenir de l'Apple IIGS.
Une pile (c'est ainsi que l'on appelle les fichiers HyperCard) a été écrite sur Mac et sur GS pour faciliter les conversions des piles entre les deux machines: il s'agit d'HyperMover.
Ne vous faîtes pas trop d'illusions: la traduction directe d'une pile Mac donne des résultats minables sur GS, à cause du manque de couleurs (et du manque de capacités musicales) des Macs pour lesquels HyperCard a été conçu.
L'intérêt de la chose est dans la possibilité de développer en parallèle, à partir de zéro, des piles pour Mac et GS, simultanément.
Ne nous trompons pas, HyperCard est un produit de qualité (par exemple, l'aide en ligne est colossale) qui va pousser les utilisateurs à gonfler leur configuration: mais vous étiez prévenus...

Super Convert (Seven Hills Software)

On se souvient de Jason Harper et de son programme SHR Convert. Il s'est aperçu que le shareware ne nourissait pas son homme, et a décidé de faire éditer commercialement une nouvelle version de son excellent programme de conversions entre de nombreux formats graphiques existant sur Apple II et Apple IIGS, MacIntosh, IBM, Atari ST, Amiga, Commodore 64/128...

Your Word Box (Word Box Company)

Programme éducatif américain; il facilite l'apprentissage de la lecture. Le logiciel est fourni sur un CD-Rom et nécessite au moins 1.5 Mo de mémoire vive. Cela permet l'utilisation intensive de dessins couleurs et d'accompagner les leçons de voix humaines digitalisées. Chaque leçon est interactive et a une durée maximum de cinq minutes.

Copy II+ 9.1 (Central Point Software)

La version 9.1 du célèbre programme de copie est disponible: des changements dans le fichier des paramètres de copie, mais je n'ai guère vu les innovations attendues...

Graphic Exchange (Roger Wagner Publishing)

La nouvelle version de Graphic Exchange (version 4.2) ne passe pas inaperçue. En effet, Graphic Exchange est maintenant une véritable application GS/OS avec menus déroulants, fenêtres, et surtout gestion standard de la mémoire.
Rappelons que cet utilitaire permet de faire de la conversion entre les différents formats graphiques de l'Apple II et du Macintosh. Il existe maintenant la possibilité de créer des procédures automatisant les conversions de plusieurs fichiers graphiques. Il est toujours possible de récupérer les images MacPaint.
Inversement pour convertir les images GS sur le Mac, pas de problème, vous trouverez dans le package une disquette Macintosh (400 Ko!!) contenant une application de conversion à vous couper le souffle...
Graphic Exchange est un utilitaire indispensable lorsque l'on manipule des images de tout type (HGR, PrintShop, News Room, DHGR, MacPaint, SHGR, DSHGR...)

My Paint (SaddleBack)

My Paint est un programme de dessin conçu pour les très jeunes enfants. Le bambin n'a pas besoin de savoir lire pour dessiner avec My Paint. L'interface utilisateur a été longuement testé. C'est aussi un livre de coloriage: sur la disquette, vous trouverez 28 images dans lesquelles l'enfant pourra dessiner et faire des coloriages; il est même possible d'ajouter des sons digitalisés.

Independance (Seven Hills Software)

Drivers pour les imprimantes Hewlett Packard Deskjet et LaserJet ou compatibles (DeskJet Plus, DeskJet 500 et LaserJet II P).

Harmonie (Vitesse)

Drivers pour les imprimantes Hewlett Packard LaserJet, Deskjet, Paintjet, ainsi que pour 24 autres imprimantes matricielles série ou parallèle (la gamme des imprimantes EPSON est bien représentée: Epson MX80, LQ, LQ 800...).

Avec Independance et Harmonie, le problème des imprimantes non Apple connectées à un Apple IIGS est maintenant résolu.

Deliverance (Vitesse)

Deliverance est un utilitaire de réparation de volumes GS/OS. Il a de plus des fonctions avancées dans la détection et la prise en compte par le système des 'bad blocks'. Il existe une commande pour la copie des fichiers, mais elle est d'une trop grande lenteur...
L'éditeur de blocs de Deliverance est séduisant. On peut soit demander la lecture d'un bloc particulier, soit tracer un fichier (commande EDIT FILE). Les fonctions classiques sont bien entendu présentes: recherche d'une chaîne en Hexa ou en ASCII, commande pour désassembler...
L'édition d'un octet est facilitée par la possibilité d'effectuer une opération logique directement en entrée (finie de jouer avec la calculette...): remplacement, addition, soustraction, négation, ET logique, OU inclusif, OU exclusif, NON logique, possibilité de jouer individuellement sur les 8 bits d'un octet, mise à zéro, décalage à gauche, décalage à droite... et cela sur un octet, un mot, un mot long ou un double mot long.
Le désassembleur permet de jouer sur la taille de l'accumulateur et des registres d'index. Les changements apportés à un bloc peuvent être temporaires... très précieux pour les maladroits.
Le nombre d'essais de lecture d'un bloc est paramétrable.
La fonction EDIT FILE INFO est particulièrement intéressante. Tous les paramètres du fichier sont indiqués et sont modifiables: Type, Type auxiliaire, Type d'enregistrement (avec Pop-up Menu pour le choix), Pointeur clé, Bloc d'index, Nombre de blocs utilisé, Date et Heure de création, les différents bits d'accès sont indiqués (lock, invisible, lecture-écriture, backup bit...). Tout cela pour la Data fork et la Resource Fork.
A quoi sert donc qu'un éditeur de blocs ait l'interface "Desktop", direz-vous? Le mode texte suffit largement pour...
STOP! Vous alliez dire une bêtise: ça sert à copier/coller et à couper/coller, bien sûr. Directement du bloc $3AF dans le traitement de textes d'Apple-

works-GS, et vice-versa, ça n'est pas intéressant, ça? Deliverance plus complet que ProSel 16? à suivre.

Disk Access (Seven Hills Software)

Ce NDA permet de faire des manipulations sur des volumes ou des fichiers sans pour autant quitter l'application dans laquelle on travaille. Cela évite de revenir au Finder ou à ProSel 16 (d'ailleurs un mini sélecteur est fourni avec Disk Access). Toutes les fonctions classiques sont présentes: Copy, Verify, Move, Check Drives, Rename, Eject, Delete, Erase, Initialize, New Folder, Item Info. Il est aussi possible de rechercher un fichier (Find File) et d'afficher le contenu d'un fichier (Show File).

Diversi Cache - Diversi Copy Diversi Key (Diversi Software)

La série des Diversi (programmée par Bill Basham) revient en force avec des versions qui fonctionnent avec le système GS/OS 5.0.
Diversi Cache (version 3.0) s'installe dans le fichier ProDos et permet d'accélérer considérablement les accès disque. On peut régler la taille du cache et les lecteurs que l'on veut accélérer en utilisant un CDA qui s'installe au moment du chargement de ProDos.
Diversi Copy (version 4.0) est une série de copieurs pour disquettes 5.25 et 3.5.Une originalité: il est maintenant possible d'installer le copieur de disquettes 3.5 en accessoire de bureau de type CDA, ce qui est très pratique.
Diversi Key permet d'ajouter des macro-commandes à vos différents programmes.
Signalons aussi la présence de Diversi Hack (version 1.6b), un CDA qui permet de passer en moniteur à tout moment et de faire une hard copy de l'écran texte sur Image Writer. Personnellement, je lui préfère The DeskTop Manager (TDM).

Close View (Apple Computer)

Cet accessoire de bureau (il s'installe en fait comme une Init) permet d'agrandir le contenu de l'écran graphique. Le facteur d'aggrandissement peut être de 2, 3, 4, 6, 8 ou 12.
Il existe plusieurs équivalents clavier pour connecter ou déconnecter cet accessoire, pour augmenter ou diminuer le facteur de grossissement et pour inverser l'affichage. Ces équivalents sont des plus utiles... vous comprendrez quand vous devrez trouver une case de fermeture ou un Pop-up Menu sur un écran graphique de 12 fois la taille normale.
Je pense que vous avez compris que Close View est un outil qui facilite la manipulation de l'Apple IIGS pour les personnes ayant des problèmes de vue.

Easy Access (Apple Computer)

Série d'utilitaires clavier facilitant la manipulation du clavier et de la souris aux personnes ayant des difficultés à se servir de leurs deux mains. Easy Access permet aussi de déplacer le pointeur de souris de manière très précise.
Le programme est une Init: il fonctionne aussi bien sous P8 ou GS/OS et dans toutes les applications. Avec Easy Access, il est par exemple possible de taper Pomme Control Escape en se servant d'une seule touche du clavier.

Video Keyboard (Apple Computer)

Cet accessoire de bureau donne la possibilité de faire de la saisie sans avoir à utiliser le clavier. En effet, il est possible de taper sur un clavier graphique que ce soit du texte ou des touches de commande (Pomme Q - Pomme S...).

Ces trois programmes: Close View, Easy Access et Video Keyboard démontrent la capacité d'Apple à s'adapter aux marchés les plus délicats.

The Ugly Duckling (ByteWorks)

On sait que ByteWorks est le spécialiste des langages sur Apple IIGS (la série des Orca); voici leur première contribution dans le genre éducatif. Mike Westerfield (le Programmeur de ByteWorks) a écrit ce programme pour sa fille.
C'est en fait un programme éducatif permettant aux enfants d'apprendre à lire. Le logiciel raconte une histoire animée et la dit grâce à une voix digitalisée: c'est l'histoire du vilain petit canard version américaine...

Les softs: freewares / sharewares

Ubu's Tile Editor (Pangea Software)

Programme en shareware qui permet d'éditer et de créer des 'tuiles' de 8x8 pixels ou des caractères. Ce programme est utilisé par Brian Greenstone pour la programmation de ses jeux.

Sound Off (TFF Enterprises)

CDev et Init qui permettent de modifier le System Beep (beep d'erreur) et de jouer une musique ou un son au moment du chargement de GS/OS. Ce programme est en shareware. Attention, à chacune de vos erreurs le cri de Tarzan peut retentir...

Audio Zap (Ian Schmidt)

Programme en shareware. C'est un utilitaire musical: edition d'un son, choix de la vitesse d'écoute, visualisation de l'onde, couper/coller, zoom, amplification, effet d'écho, inversion... Audio Zap permet aussi l'enregistrement.
Il reconnaît les formats ASIF/SoundSmith, ACE, HyperStudio et binaire. Il gère les cartes SuperSonic, Sonic Blaster, Audio Animator ainsi que le circuit à digitaliser de HyperStudio. Audio Zap est un véritable studio d'enregistrement et de modification des sons obtenus. En prime, Audio Zap intègre un oscilloscope.
L'option OPEN DISK permet de lire les disquettes n'ayant une structure de catalogue ProDos: l'auteur a pensé aux disquettes Nucleus, Space Harrier, Modulae, ECC Demo... que l'on peut maintenant piller très simplement.
Ian Schmidt est tout à fait fréquen-

Parler en PostScript

Le GS sait envoyer un fichier PostScript à la Laser, mais si! Une fois que vous l'avez engendré avec ⌘-F, il suffit de l'appeler "IWEM" et de le mettre à la place de l'autre, et le CDEV LaserWriter l'enverra! Merci à Yvan Kœnig et à Mr Bonnet pour le tuyau.

table: il déconseille de se servir du Finder pour lancer Audio Zap.

Write It 2.0 (Ravenware Software)

NDA écrit par C. K. Haun. C'est un petit programme de traitement de texte en shareware. Cet accessoire permet de prendre quelques notes sans quitter l'application dans laquelle on se trouve.
Ce traîtement de texte est construit autour de TextEdit. Ses fonctions sont classiques: Edition, Recherche/Remplacement, Changement de police, de style et de corps. La fonction Hi Bit Settings permet de jouer sur le bit de poids fort des caractères ASCII que l'on tape: on peut forcer ce bit à 0 ou à 1 (pratique pour les gens utilisant Merlin).
Write It a lui-même un menu Pomme; il est amusant d'appeler un NDA à partir d'un autre NDA.

Tonight Sky (Air Ship Software)

Programme en shareware. C'est un programme qui transforme l'Apple IIGS en planétarium. On peut visualiser le ciel dans son ensemble ou sélectionner une partie de celui-ci. Bien entendu, la position des planètes évolue en fonction de la date.

A lire

GS + (Ego System)

Un nouveau magazine pour l'Apple IIGS. Etonnant! Il est constitué d'une revue papier et d'une disquette; son nom: GS +.

Cours de C (Byte Works)

Après le cours de Pascal (voir ToolBox Mag numéro 1), voici le "Learn to program in C". D'après Byte Works, la programmation en C à l'aide de ce livre est facile et amusante: vous apprendrez dès la première leçon à écrire un programme en mode desktop, puis vous apprendrez à écrire des applications, des CDA et des NDA.
Chaque leçon est constituée d'exemples de programmes, de problèmes et des solutions à ces problèmes. Pour les gens qui n'aiment pas faire de la saisie, pas de panique, tous les sources se trouvent sur la disquette accompagnatrice. Le livre est fourni avec le compilateur ORCA/C.

8/16 Central (Resource Central Inc.)

La revue 8/16 cesse sa publication. Ce n'est pas une disparition, mais un changement de propriétaire et de forme de diffusion. C'est maintenant Resource Central Inc. (anciennement A2 Central (anciennement Open Apple)) qui édite 8/16 Central (anciennement 8/16). 8/16 avait pour Rédacteur en chef Ross Lambert. Le Rédacteur en chef de 8/16 Central s'appelle Jay Jennings (il collaborait à 8/16).
La revue cesse d'avoir un support papier et est uniquement constituée d'une disquette; cette forme de diffusion est adaptée car la revue compte seulement, de par le monde, un millier d'abonnés. Au niveau contenu, la revue reste une publication destinée aux programmeurs sur Apple II et IIGS.

Fausse vierge

Quelques lecteurs nous ont fait remarquer qu'il était *"plutôt cher de facturer 100 F la disquette Bis, alors que la disquette vierge ne coûte pas plus de 5 F"*. C'est parfaitement exact.
Utilisée comme disquette vierge, la disquette 2 Bis n'est pas seulement un vol manifeste sur le prix, c'est aussi une tromperie sur la marchandise: en effet, ***cette disquette a déjà servi. Elle n'est même pas vierge: il y a des programmes dessus!***

ToolBox Mag

Fontes ImageWriter, suite (3)

Cette fois, il y avait un peu de place sur la disquette pour mettre nos "vraies fontes ImageWriter" (sur cette notion, et sur l'utilisation de ces fontes, voir ToolBox Mag n° 1).

Dans le sous-catalogue /JYB.FONTES, vous trouverez une série de fontes appelée "**Saint.Denis**" (eh non, Eric, toujours pas de fontes Argenteuil...). Le dessin de ces fontes est inspiré de "Times". Mais attention, contrairement au Times de la Laser, qui sert à imprimer le texte courant, Saint-Denis ne donne qu'un résultat médiocre dans les petites tailles (fontes 10/20).
Cette fonte donne de bons résultats pour faire des **titres**. J'écris couramment en Bobigny 12, avec les notes en Bobigny 9 et les titres en Saint.Denis.

Vous en aurez la preuve avec les **deux modèles de jaquettes de cassette audio** (fichier PAO AppleWorks-GS), que vous trouverez dans le catalogue /JAQUETTES.K7.
Le premier, intitulé "JAQ.COND.SUP", est conçu pour être imprimé sur Imagewriter en modes Condensé (Condensed) et Qualité Supérieure (Better Text). Le second est intitulé "JAQ.COND.50", et il est conçu pour être imprimé sur Imagewriter en modes Condensé et Qualité Normale (Better Color), mais avec une réduction de 50% (cette seconde formule permet à Laurent Bourdin de mettre plus facilement une image de la tête de Bob Marley).

Bien entendu, il faut pour cela que l'ensemble des fontes Bobigny de ToolBox Mag 1, et l'ensemble des fontes Saint-Denis de ToolBox Mag 3, soient placées dans votre catalogue /SYSTEM/FONTS.

Note: je me suis trompé à propos du vrai driver de LQ du système 5.03. Ce sont des fontes de taille x 3, et non des fontes de taille x 4, qu'il utilise. Si vous avez une LQ, bon courage pour vous fabriquer ces fontes!

JYB

Essai logiciel ():

Platinum Paint

Eric Weyland

Le tout dernier logiciel de dessin de Beagle Bros a déjà une longue histoire... Elle commence en Octobre 1986, avec la naissance de l'Apple IIGS. C'est à cette époque que l'auteur, Matthew Reimer, commence la programmation de ce qui allait devenir Platinum Paint. En 1988, Jem Software commercialise pour $25 MiniPaint, l'ancêtre de Platinum Paint.

Platinum Paint reprend les caractéristiques les plus puissantes de PaintWorks Gold, DeluxePaint II et du programme de dessin PixelPaint du Macintosh II; Platinum Paint intègre des fonctions originales dont nous reparlerons plus loin...

Configuration nécessaire:
- Apple IIGS couleur avec 1 Mo de mémoire vive
- Lecteur de disquettes 3.5"
- Système d'exploitation GS/OS (des bribes du système GS/OS 5.03 sont livrées avec Platinum Paint)

Platinum Paint n'est pas protégé contre la copie. Vous pouvez l'installer sur un disque dur; le programme est au standard GS/OS.

Comme tous les programmes de dessin, Platinum Paint va nous permettre de travailler au moyen d'une boîte à outils située à gauche de l'écran. Cette boîte à outils peut se placer n'importe où sur l'écran.

Les outils permettent de créer différents objets et de sélectionner différentes parties de votre image en conjonction avec d'autres commandes.

On se rend compte rapidement qu'un des points forts de Platinum Paint est **la richesse qu'il offre dans les équivalents clavier** et en fait dans **la possibilité de tout faire par l'intermédiaire du clavier**. En effet, le curseur de la souris se déplace lorsqu'on utilise les flèches directionnelles; le click de la souris peut se faire avec la touche Return. On peut aussi choisir au clavier l'affichage des coordonnées cartésiennes du curseur, la couleur à utiliser, la taille de la brosse... On peut par la touche Esc masquer le curseur pour voir précisément ce que l'on fait.

Les touches Pomme, Option, Control et Shift ajoutent des options lorsque l'on s'en sert avec les outils. Ainsi, la touche Shift force un tracé à être parfaitement horizontal ou vertical. En mode 640, certains outils agissent d'une façon différentes selon que l'on se trouve en mode Dither ou non (En mode graphique 640, seules quatre couleurs pures sont disponibles. Le Dithering est une technique utilisée pour obtenir 16 couleurs à partir des quatre couleurs de base. Quand deux pixels de différentes couleurs sont côte à côte, ils sont considérés comme un pixel unique utilisant un mélange des deux couleurs).

Les outils sont très classiques: le Marqueur, le Lasso, la Main, le Texte, le Pot de peinture, le Vaporisateur, la Brosse, le Crayon, la Ligne, la Gomme, le Rectangle, le Rectangle arrondi, le Rond, le Courbe, la Forme libre, et le Polygone. J'oubliais le fameux 'Dropper' qui n'a pas d'icône dans la boîte à outils. On le sélectionne à partir du clavier. Je vous laisse lire la documentation pour savoir comment... Cet outil permet de changer la couleur du crayon simplement en cliquant sur un pixel du document. Le crayon prend alors la couleur de ce pixel.

Les **options associées aux outils** sont très variées.

Par exemple, lorsqu'on sélectionne une portion d'image avec le marqueur, il est possible d'ajuster la boîte de capture à la portion d'image sélectionnée. On peut faire une sélection en dehors de la fenêtre affichée à l'écran; c'est en fait un **scrolling de la sélection**: dans la plupart des autres programmes, la sélection ne peut se faire que sur une portion de dessin affichée à l'écran.

En mode Lasso, on fait une sélection d'une partie d'image sans tenir compte de son fond. Ceci permet de récupérer facilement les dessins à formes complexes. Les pixels du dessin se trouvant sous le Lasso peuvent être pris en compte. On peut aussi sélectionner la forme et son fond graphique.

Dans le tracé des différentes formes (rectangle, cercle), on peut forcer l'absence de contour. Cela

n'est évidemment utile que pour les formes pleines.

Toutes les formes que l'on trace peuvent être ombrées (y compris le texte et les lignes).

Souvent, une fois une portion d'image sélectionnée, on peut la transformer en utilisant les commandes de la barre de menus. Ainsi, il est tout à fait possible d'enlever ou de remplacer une couleur particulière, de faire de nombreuses distorsions et transformations: retournement horizontal ou vertical, effet miroir horizontal ou vertical, réduction, agrandissement, choix de l'échelle... La commande **Rotate** permet de définir un angle de rotation pour la zone sélectionnée, l'effet **Slant** permet une déformation dans un sens simplement en tirant sur une des poignées de la boîte de capture, la commande **Perspective** permet une déformation dans tous les sens.

Dans Platinum Paint, on peut avoir **quatre document ouverts simultanément.** Ils se placent alors dans des fenêtres différentes. Lorsque l'on charge une image, il est possible de la placer dans la fenêtre courante en remplacement du dessin qui s'y trouve. Sinon, une nouvelle fenêtre est ouverte.

Au moment de la sauvegarde vous avez le choix entre le format Apple Preferred et le format SHR (fichier écran non compressé). Platinum Paint, c'est là une **lacune importante**, ne gère pas les fichiers graphiques de **plusieurs écrans de large** et de plus de deux écrans de long.

Au niveau de l'importation des différents types de fichiers graphiques, Platinum Paint se débrouille plutôt bien. Vous avez la possibilité d'importer des images HGR, DHGR, PrintShop, PrintShop GS ou MacPaint (les images MacPaint doivent d'abord être converties au format ProDos avec A.F.E.).

Platinum Paint possède **toutes les fonctions maintenant classiques** d'un logiciel de dessin. On y trouve bien entendu des possibilités d'animation (format PaintWorks Gold), la possibilité de fabriquer ses propres brosses à partir de n'importe quelle partie d'image (possibilité de sauvegarde), le zoom (effet de loupe), rotation des couleur, masques (protection d'une zone graphique ou de couleurs)...

Les **originalités** de Platinum Paint sont nombreuses. D'abord la gestion et la manipulation des palettes de couleurs sont séduisantes. Lorsque l'on édite une palette, on a toujours à l'écran une portion de l'image sur laquelle on travaille (avec une possibilité de scrolling), ce qui facilite grandement la localisation d'une couleur dans la palette. Diverses palettes de couleurs sont disponibles: la palette par défaut (palette système utilisée au moment de la création d'un nouveau document), une palette constituée uniquement de nuances de gris, puis trois autres palettes (EarthTone, Rainbow et Pastels). De façon très classique, les couleurs sont modifiables grâce à des barres de défilement qui permettent d'ajuster les composantes Rouge, Verte et Bleue de chacune d'entre elles. Platinum Paint permet aussi d'utiliser le système HSV qui décompose une couleur en trois constituantes: Hue (en français teinte), Saturation, et Valeur système. La documentation de Platinum Paint (au passage je signale qu'elle est très bien faite) détaille le système HSV. Au niveau de la palette, on peut faire des couper/coller entre les différentes couleurs, les éclaircir, les foncer, ajouter du Rouge pour faire plus chaud (commande Warmer), refroidir en enlevant du Rouge (commande Cooler), faire des mélanges de couleurs...

La **gestion des masques** est particulièrement bien réalisée. Un masque permet de protéger une zone graphique ou une (ou plusieurs) couleur(s) dans un document. Ainsi cette zone ne peut plus être modifiée (sauf si on supprime le masque). On peut ajouter au masque différentes zones graphiques puis les retirer, y ajouter ou enlever différentes couleurs, inverser le masque, faire clignoter le masque, supprimer le masque...

Une **page de brouillon** (Spare Page) permet de faire d'importantes modifications sans avoir à faire des acrobaties dans les sauvegardes. Il est bien entendu possible de copier le document dans cette page de brouillon et inversement.

Platinum Paint est le seul logiciel de dessin sur Apple IIGS à tracer des **courbes de Bézier.**

Les commandes et équivalents clavier sont omniprésents (très bon résumé en fin de documentation).

En conclusion, outre les supériorités techniques de Platinum Paint sur ses concurrents PaintWorks Gold et DeluxePaint II, il est important de souligner que ces derniers ne sont plus suivis par leur éditeur.
Platinum Paint tourne avec le système GS/OS sans aucune modification; on se rappelle les fantaisies de PaintWorks Gold dans la gestion de la mémoire et l'allergie de DeluxePaint II au GS/OS.
Le seul manque grave de Platinum Paint est l'impossibilité de travailler avec des fichiers QuickDraw de taille importante.

Courrier des lecteurs

Robert Santelli: Graphic Writer 3 a-t-il les lettres accentuées françaises comme AppleWorks-GS?

ToolBox Mag: Oui. Le FrInit et le Sys.Resources d'Yvan Kœnig de notre numéro 2, ainsi que notre article-tableau de ce numéro contribuent à régler la question pour tous les programmes "réellement GS", comme Graphic Writer 3 et AppleWorks-GS.

Robert Santelli: Comment faire un fichier A4 de huit écrans avec la PAO d'AppleWorks GS?

Toolbox Mag: Créez à partir de zéro un fichier "Page Layout" dans AppleWorks-GS. Passez dans le menu "Page Setup", et choisissez "Condensé Vertical" et "Réduction 50%". Regardez ensuite votre page: elle fait bien 2 écrans de large sur 4 de haut. Et pourtant elle s'imprimera sur une page A4.

Robert Santelli: Plutôt que de photographier l'écran, j'aimerais pouvoir gérer une vraie imprimante couleur, comme la HP PaintJet. Le GS le mérite.

ToolBox Mag: Voyez Vitesse (Harmonie) et Seven Hills (Independance). Ils vendent tous les deux les drivers nécessaires. Si vous avez une HP PaintJet, achetez les deux drivers, comparez, et *faîtes un article pour ToolBox Mag*, merci !

Norbert Thiébaud: Utilisateur passionné d'Apple depuis plus de 7 ans, j'ai malheureusement dû limiter mes activités en ce domaine ces deux dernières années, pour des raisons d'études. Mon archaïque version d'APW refuse de lire certains fichiers de la disquette ToolBox Mag. Qu'est-ce qu'un fichier de "storage type 5"?

ToolBox Mag: Deux ans, c'est beaucoup, sur GS, mais vous avez une excuse valable. Les "fichiers du cinquième type" sont des fichiers avec Data Fork et Resource Fork de GS/OS. Ils contiennent généralement des ressources. Voyez la documentation de GS/OS 5.02 français, le GS/OS Reference Manual, APW version 1.1, et surtout... ToolBox Mag pour les ressources.

Pas le même bureau

S'il vous plaît, ne mélangez pas dans votre courrier ce qui concerne ToolBox Mag et ce qui concerne les autres produits ToolBox.
Si vous profitez du même envoi, mettez deux feuilles ou deux disquettes différentes.
Si vous demandez, dans une lettre adressée à ToolBox Mag, des feuilles d'errata pour le livre GS Epluché, vous ne recevrez jamais vos errata, car votre demande ne sera pas transmise.
Eh oui, quand ça grossit, ça se bureaucratise... Merci d'en tenir compte!

Le monde à l'envers !

Sous prétexte que le GS ressemble de plus en plus à un Mac, des programmeurs cherchent à introduire sur le GS ce que j'ai appelé "l'esclavage Mac", à savoir qu'on ne donne pas de choix à l'utilisateur, on choisit à sa place.
Prenons l'exemple de la **commande Quit**: sur le Mac, à la différence du GS (voir ToolBox-Mag 2 page 41), le Finder ne se quitte pas, on ne le quitte qu'en rebootant. Comme GS/OS a introduit une commande "ShutDown", on voit de plus en plus de lanceurs de programmes GS qui n'acceptent pas de quitter comme tout le monde, d'être lancés depuis un autre lanceur. Ils forcent un ShutDown, c'est-à-dire un reboot.
C'est particulièrement évident avec **Wings**, le lanceur de programmes de Vitesse. J'ai lancé les trois versions de Wings: cela m'a forcé à rebooter trois fois! Autant vous dire qu'il n'y aura pas de quatrième: **Wings n'est pas au standard GS**.
Et ne mélangeons pas tout: si Photonix II nous force à rebooter en sortant, c'est parce qu'il ne **peut** pas faire autrement (sur les GS 1,2 mégas, il a été obligé de voler la mémoire du système). Mais Wings, c'est différent: lui, il ne **veut** pas!
Apple, qui connaît son monde, a pourtant clairement prévenu les développeurs de ne pas faire ça. L'ordinateur qui choisit à notre place, c'est **le monde à l'envers**: nous ne l'accepterons pas!
JVB