Atari ST : menu et compilation

Frédéric Sagez
Codeur du groupe
NoExtra-Team
fsagez@gmail.com

Vous avez eu un ATARI ST et vous avez déjà utilisé des compilations de groupes comme FUZION, ELITE, ZUUL ou AUTOMATION, alors cet article tombe bien car il vous explique comment on fabrique ce type d'interface. Et le tout en assembleur s'il vous plait ! Aujourd'hui je vais vous expliquer comment faire une compilation d'intros tout en utilisant différents outils tel que Néochrome Master pour la partie graphique, le GFA BASIC pour générer nos données brutes telles que la fonte ainsi que l'assembleur DEVPAC pour compiler nos programmes.

Comme indiqué dans le numéro 203 de janvier, nous allons faire une incursion dans le monde de la compilation. Une compilation ou menu contient des éléments légaux et/ou illégaux, cela va du jeu ou de la version crackée (vies infinies ou déplombé), de démonstrations, ou encore de logiciels, voire de sharewares suivant le support proposé. Le menu est souvent interactif, on sélectionne l'item que l'on veut voir via la souris ou le clavier et hop c'est parti !

Comment ça marche ?

C'est assez simple : nous avons besoin d'un Loader qui va se charger d'exécuter autant de programme que l'utilisateur le désire via un Menu. En fait le Loader est agencé comme ceci : une fois lancé, il va afficher une image ou des informations, ensuite il va charger en mémoire et exécuter le Menu. Généralement bien compressé à cause de la taille du fichier, il s'exécute assez rapidement lors du chargement de celui-ci via la disquette. A partir du Menu, une fois que l'utilisateur a choisi ce qu'il veut voir, il va renvoyer en mémoire la sélection de l'utilisateur et rendre la main au Loader qui va se charger d'exécuter le programme sélectionné et ensuite de recharger le Menu une fois que l'utilisateur aura fini de regarder le programme. La sortie du Menu et le retour au bureau GEM sont prévus lorsque vous cliquez sur la touche ESCAPE dans le cadre de notre démonstration. La figure [1] vous explique le principe du fonctionnement continu du chargement et de l'exécution des programmes.

Le LOADER

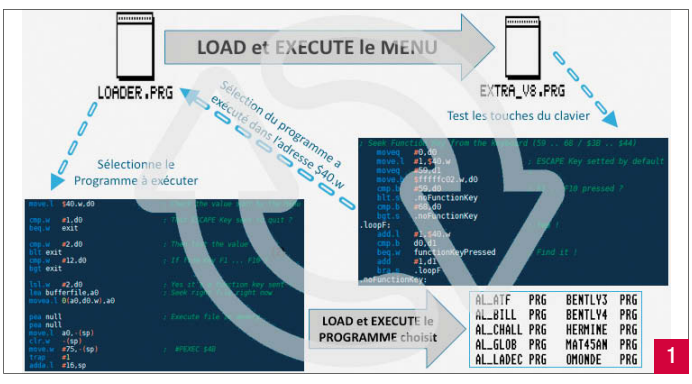
Le Loader est le fichier de chargement, il permet de charger un ou plusieurs fichiers en mémoire et de l'exécuter ensuite. Pour s'assurer que le Loader s'exécutera à chaque fois que l'on insère la disquette et qu'il sera exécuté automatiquement au démarrage, on créera un répertoire **AUTO** à la racine du lecteur A et on le copiera le fichier *LOADER.PRG* à l'intérieur.
A chaque action de chargement, nous afficherons une information. Au premier démarrage de celui-ci on affichera le Logo **[2]** du groupe NoExtra-Team, puis pour chaque chargement de programme, nous indiquerons LOADING en blanc ou ERROR en rouge si il y a eu une erreur au chargement du fichier.
Pour charger nos programmes, nous utiliserons la fonction **Pexec()** de l'Atari ST, c'est une fonction **GEMDOS** qui permet de charger des programmes à partir de la disquette. Nous lui passons 4 paramètres via la pile mais seuls 2 paramètres sont utiles pour notre Loader : le nom du fichier à charger ainsi que le mode de chargement. Celui-ci est à zéro, soit en mode charge et exécute. Nous utilisons le code retour de cette fonction qui est mis dans le registre de données D0 afin de vérifier s'il y a eu un problème lors du chargement d'un fichier, ceci afin d'indiquer une erreur à l'utilisateur. La gestion de l'erreur de chargement d'un fichier nous assure de toujours revenir au Menu afin de ne pas laisser l'utilisateur sur un écran final décoré de jolies bombes.

Le MENU

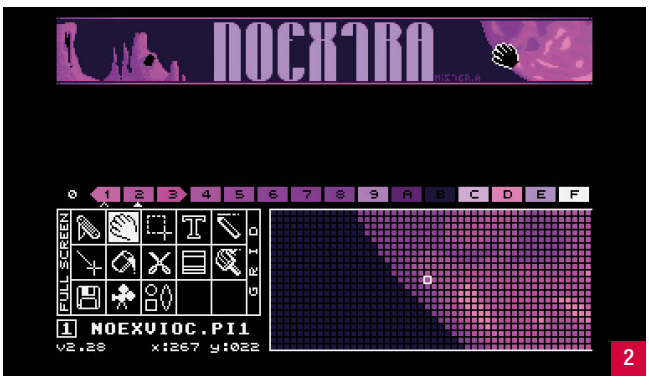
Le Menu est le programme qui permet de sélectionner et de choisir ce que l'on veut voir à partir de celui-ci. Dans notre cas, nous allons afficher du texte qui permettra de sélectionner des intros que voudra visionner l'utilisateur. A part la partie graphique, sonore et visuelle, chaque menu est composé généralement de textes, indiquant des informations relatives à son contenu ainsi qu'un scrolltext pour faire passer des messages à différentes personnes, groupes, etc.
Le menu va contenir principalement :

- Un texte proposant le contenu du Menu,
- Une animation avec un effet 3D,
- Un scrolltext contenant des messages, de la pub, etc. ,
- Et de la musique : un module Amiga 4 voies sera joué en fond.

Une interaction avec les touches de Fonction de F1 à F10 du clavier du ST sera géré pour sélectionner l'introduction à visualiser. La touche Escape permettra de sortir du menu et de



Description du fonctionnement du Loader



Logo 16 couleurs en 320x40 par Mister.A

retourner sur le bureau GEM. Le Menu pourra fonctionner soit avec un Atari STF ou STE avec 512 ko de mémoire grâce à la détection de la machine et du Player de module utilisé. [3]

Pour créer le menu, nous utiliserons notre propre Framework qui se trouve à l'adresse suivante https://github.com/NoExtra-Team/framework et nous rajouterons nos effets d'affichage de textes, de scrolltext, effet 3d affichant un cercle sur deux plans et une musique de type Module 4 voies. Et le plus important : nous allons gérer les touches de fonctions de F1 à F10 de l'Atari ST, nous détecterons quelle touche sera appuyée pour la sélection du programme à visualiser et nous retournerons le numéro du programme à exécuter via le Loader tout en indiquant cette information en mémoire à l'adresse $40.w.

Le code Assembleur

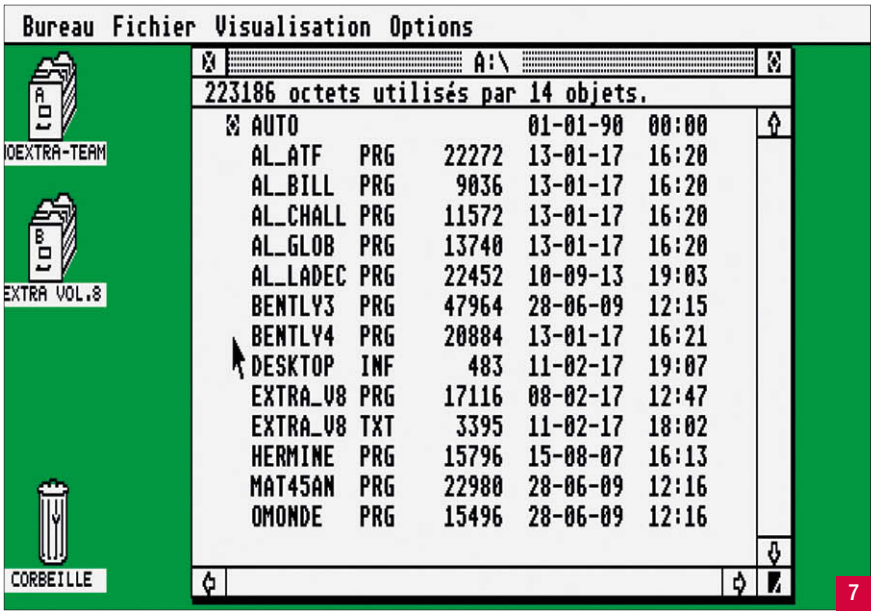
La fonte qui servira pour le texte et le scrolltext sera créée avec un logiciel de dessin [4]. Nous utiliserons une fonte un plan - soit une couleur - de taille 8 par 8 pixels provenant du jeu Bobble Bubble. Notre fonte commence toujours par un ESPACE et finit par la lettre minuscule 'z'. L'ESPACE correspondant au caractère hexadécimal $20 et la lettre 'z' correspondant au caractère hexadécimal $7A, la chaîne de caractères ainsi affichée correspondra avec les caractères standard ASCII utilisés par le ST. En fait, on fera correspondre la chaîne de caractères à afficher via les codes hexadécimaux des caractères ASCII. Exemple, pour afficher le caractère majuscule 'A', correspondant au code hexadécimal $41, nous afficherons le 33ème caractère sur la première ligne de l'image de la fonte. (33 en décimal correspondant à 21 en hexadécimal plus 20 comme valeur de base - c'est le caractère ESPACE – qui donnera 41 en hexadécimal)

Nous utiliserons le **GFA BASIC** qui est le Basic standard sur Atari ST pour transformer notre image Degas *BOBBLE.PI1* en un fichier binaire qui sera utilisé dans le Menu. Le programme *CODE8_8.LST* permet de générer un fichier de type RAW de la fonte de taille 8 pixels sur les 3 lignes de l'image Degas. [5]

Dans le code de notre menu, fichier *EXTRA_V8.S*, la routine *print_text* va servir à imprimer le texte du menu. Cette routine est dédiée aux résolutions Basse ou Moyenne nécessitant un plan pour afficher les caractères et pourra utiliser des fontes de taille 8 pixels ou 6 pixels. (La taille réelle de la fonte utilisée est de 8x8 pixels). [6]

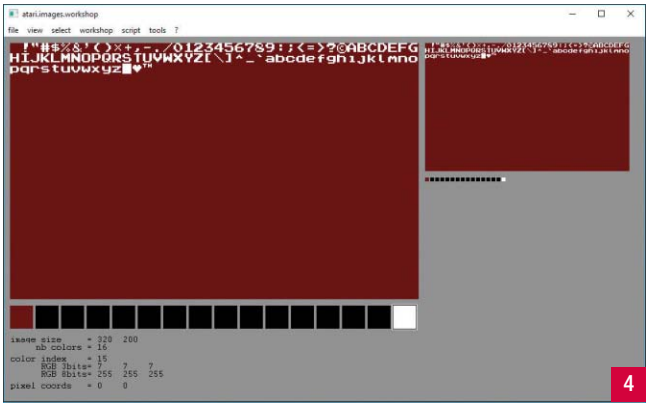
La routine scrolltext utilise aussi la fonte générée par le code écrit en GFA-BASIC. Tout comme l'affichage du texte, nous prenons le code hexadécimal du caractère, on le multiplie par 8 - taille de la fonte en largeur - puis on le soustrait à 256 pour pointer directement sur le caractère au format binaire issu des données sur l'écran final. Par contre pour le scrolltext nous passerons par un buffer pour gérer le décalage à gauche de notre chaîne de caractères puis nous l'afficherons sur l'écran physique à la position et au plan désiré. Concernant la routine 3d, nous afficherons un peu plus de 400 points sous la forme d'une sphère dont nous allons calculer la courbe et la forme dans l'initialisation via la routine *init_sphere*. Suivant la position du point et suivant l'axe des X, nous afficherons des points de couleurs bleu et violet pour indiquer le fond de la sphère afin de lui donner de la profondeur via la routine *play_sphere_3d*.

Au niveau sonore, nous utiliserons une routine qui permet de rejouer des modules 4 voies via le TIMER A et avec le TIMER D suivant le type de l'Atari ST détecté par la routine *Play__Module*. La routine Soundtrack du groupe **WizzCat/Delta Force** permet de jouer soit sur un Atari STF via le composant YM2149 sur 3 voies Mono ou sur Atari STE via le composant DMA permettant une meilleure sonorité : 4 voies Stéréo et une meilleure gestion des



Contenu de la Disquette

Menu de la compilation EXTRA



Fonte 8x8 pixels 1 plan du jeu Bubble Bobble

```
691  ***************************************************************
692  ***                DISPLAY TEXT FONT 8x8 or 6x6               ***
693  ***            MED OR LOW RESOLUTION - ONE BITPLANE           ***
694  ***                    ZORRO2/NOEXTRA-TEAM                    ***
695  ***************************************************************
696  CHARS      EQU 40  ; chars per line, 80=for med res, 40 for low res *
697  LINES      EQU 33  ; 33 for 8x8 font, 45 with 6x6 font              *
698  FONTSIZE   EQU 8   ; 8=8x8, 6=6x6 font                              *
699  SHIFTSIZE  EQU 4   ; 2=MED RESOLUTION, 4=LOW RESOLUTION             *
700  POS_SCR    EQU 160*46 ; Top Position in the Screens                 *
701  PLAN_SCR   EQU 4   ; Bitplane used to display the text              *
702  ***************************************************************
703  print_text:     lea     message,a2          ; Read the text or hexa
704  new_char:       bsr     _x_conversion       ; Convert / Resolution
705                  moveq   #0,d0
706                  move.b  (a2)+,d0            ; If zero, stop routine
707                  cmp.b   #0,d0
708                  beq     line_feed           ; Next line / RC
709  .test_plan_1:   cmpi.b  #$fd,d0             ; Bitplane + 0
710                  bne.s   .test_plan_2
711                  move.w  #0,pointeur_plan
712                  bra.s   new_char
713                  bra.s   .end_of_line
714  .test_plan_2:   cmpi.b  #$fc,d0             ; Bitplane + 2
715                  bne.s   .end_of_line
716                  move.w  #2,pointeur_plan
717                  bra.s   new_char
718  .end_of_line:   cmpi.b  #$ff,d0             ; End of the text ?
719                  bne.s   process_char
720                  rts
721
722  process_char:   asl.w   #3,d0               ; Find ASCII value
723                  lea     fonts,a1            ; and choose character font
724                  sub.w   #256,d0
725                  adda.w  d0,a1
726                  moveq   #0,d1               ; Position of the letter
727                  add.w   pointeur_plan,d1    ; in the two Screens
728                  add.l   y_offset,d1
729                  add.l   x_offset,d1
730                  movea.l physique,a0         ; In First screen
731                  lea     POS_SCR+PLAN_SCR(a0),a0
732                  adda.l  d1,a0
733                  movea.l physique+4,a3       ; In Second screen
734                  lea     POS_SCR+PLAN_SCR(a3),a3
735                  adda.l  d1,a3
736                  rept FONTSIZE               ; Print letter on Screens
737                  move.b  (a1),(a0)
738                  move.b  (a1)+,(a3)
739                  lea     160(a0),a0
740                  lea     160(a3),a3
741                  endr
742                  addq.w  #1,x_curs
743                  cmpi.w  #CHARS,x_curs       ; 79 for MED res
744                  bls     new_char
745                  move.w  #CHARS,x_curs       ; 79 for MED res
746                  bra     new_char
747
748  line_feed:      clr.w   x_curs              ; back to first char
749                  addi.l  #FONTSIZE*160+160,y_offset ; when u reached ',0'
750                  cmpi.l  #LINES*FONTSIZE*160,y_offset
751                  bls     new_char
752                  move.l  #LINES*FONTSIZE*160,y_offset
753                  bra     new_char
```

6

Le code source en assembleur 68k

```
1   REM ------------------------------------------------
2   REM --- Encodage fonte 8x8 pixels in one bitplane ---
3   REM ------------------------------------------------
4   ' Initialisations
5   RESERVE 10000
6   buffer%=MALLOC(2*8*16*3)
7   buf%=buffer%
8   width&=8
9   height&=8
10  ' Charge le dessin de la fonte (PI1)
11  CLS
12  FILESELECT #"PICTURE","C:\MENU\font\*.PI1","",fontpic$
13  BLOAD fontpic$,XBIOS(2)-34
14  ~XBIOS(6,L:XBIOS(2)-32)
15  ' Encodage de la fonte
16  '  --- Ligne 1 ---
17  FOR n&=0 TO 152 STEP width&
18    adr%=XBIOS(2)+n&
19    adr2%=adr%+1
20    @encodage
21  NEXT n&
22  '  --- Ligne 2 ---
23  FOR n&=0 TO 152 STEP width&
24    adr%=XBIOS(2)+height&*160+n&
25    adr2%=adr%+1
26    @encodage
27  NEXT n&
28  '  --- Ligne 3 ---
29  FOR n&=0 TO 152 STEP width&
30    adr%=XBIOS(2)+2*height&*160+n&
31    adr2%=adr%+1
32    @encodage
33  NEXT n&
34  ' Sauvegarde de la fonte
35  BSAVE "C:\MENU\font\FONT8_8.DAT",buffer%,2*8*16*3
36  ~MFREE(buffer%)
37  EDIT
38  ' Procdeure Encodage
39  PROCEDURE encodage
40    FOR t&=0 TO height&-1
41      POKE buf%,PEEK(adr%)
42      ADD buf%,1
43      ADD adr%,160
44    NEXT t&
45    FOR t&=0 TO height&-1
46      POKE buf%,PEEK(adr2%)
47      ADD buf%,1
48      ADD adr2%,160
49    NEXT t&
50  RETURN
```

5

Génération de la fonte au format binaire

effets. Vous pouvez télécharger des modules via les sites **Mod Archive** (https://modarchive.org/) ou **Amiga Music Preservation** (http://amp.dascene.net/) si vous voulez utiliser votre Soundtrack préféré.

Les Programmes

Assurez-vous bien que les programmes utilisés rendent toujours bien la main une fois exécutés sous le **GEM**. (Portant l'extension « .PRG »)

Pour un meilleur chargement via la disquette, n'hésitez pas à les compresser avec un outil tel qu'**UPX**, c'est un compresseur puissant multi-plateforme qui compresse bien les exécutables Atari comme sur PC. (https://upx.github.io/)

Dans le menu que je vous présente, huitième édition déjà ! Il sera composé d'introductions faites pour la Team *Atari Legend* lors de la diffusion de jeux sur Atari ST via leur site Internet http://www.atarilegend.com/.

Comme on le voit dans **[7]**, le répertoire **AUTO** contient le Loader qui sera exécuté au boot de la disquette ainsi que tous les programmes qui seront appelés par le Menu qui est le programme *EXTRA_V8.PRG*.

Les Sources

Tous les sources sont disponibles sur le **Github** de NoExtra-Team à l'adresse suivante : https://github.com/NoExtra-Team/framework/sources/menu.

Il contient 3 répertoires :

- Le répertoire /font contient les données pour encoder la fonte (routines écrites en GFA-BASIC plus le fichier Degas de la fonte utilisée) ;
- Le répertoire /intros contient toutes les introductions de jeux contenues dans la compilation ;
- Le répertoire /loader contient tout le code source du LOADER (routine assembleur plus les datas graphiques) ;
- Le répertoire /menu contient tous les éléments pour compiler le menu (routines assembleurs - core + les effets + replayer de module - avec les datas).

Ressources

Dead Hacker Society, le site Internet de la démoscène ATARI : http://dhs.nu/

Plein de sources assembleurs tout azimut sur http://freddo.chez.com/Sources/Sources.html

Pour les Cours en assembleur 68000, je vous conseille ceux fait par Féroce Lapin. Ils sont visibles en ligne sur http://atariste.free.fr/asm/assembleur.html ou consultable au format PDF sur http://sasfepu78.fr/articles/Cours_asm_68000_Feroce%20Lapin.pdf

Le référentiel Github de sources de groupes connus sur Atari ST : https://github.com/ggnkua/Atari_ST_Sources

Sinon tous les projets Atari sont référencés sur le site The Orphaned Projects Page sur http://www.ataricq.org/.

Petit oubli de ma part dans mon dernier article : les fichiers musicaux de type Soundchip sont disponibles sur le site YM2149 Archive de *Phil Graham* à l'adresse suivante : http://sndh.atari.org/.

Le site Web regorge en effet de toutes les musiques 3 voies de jeux ou de démos spécifiques au composant Yamaha de l'Atari ST. •