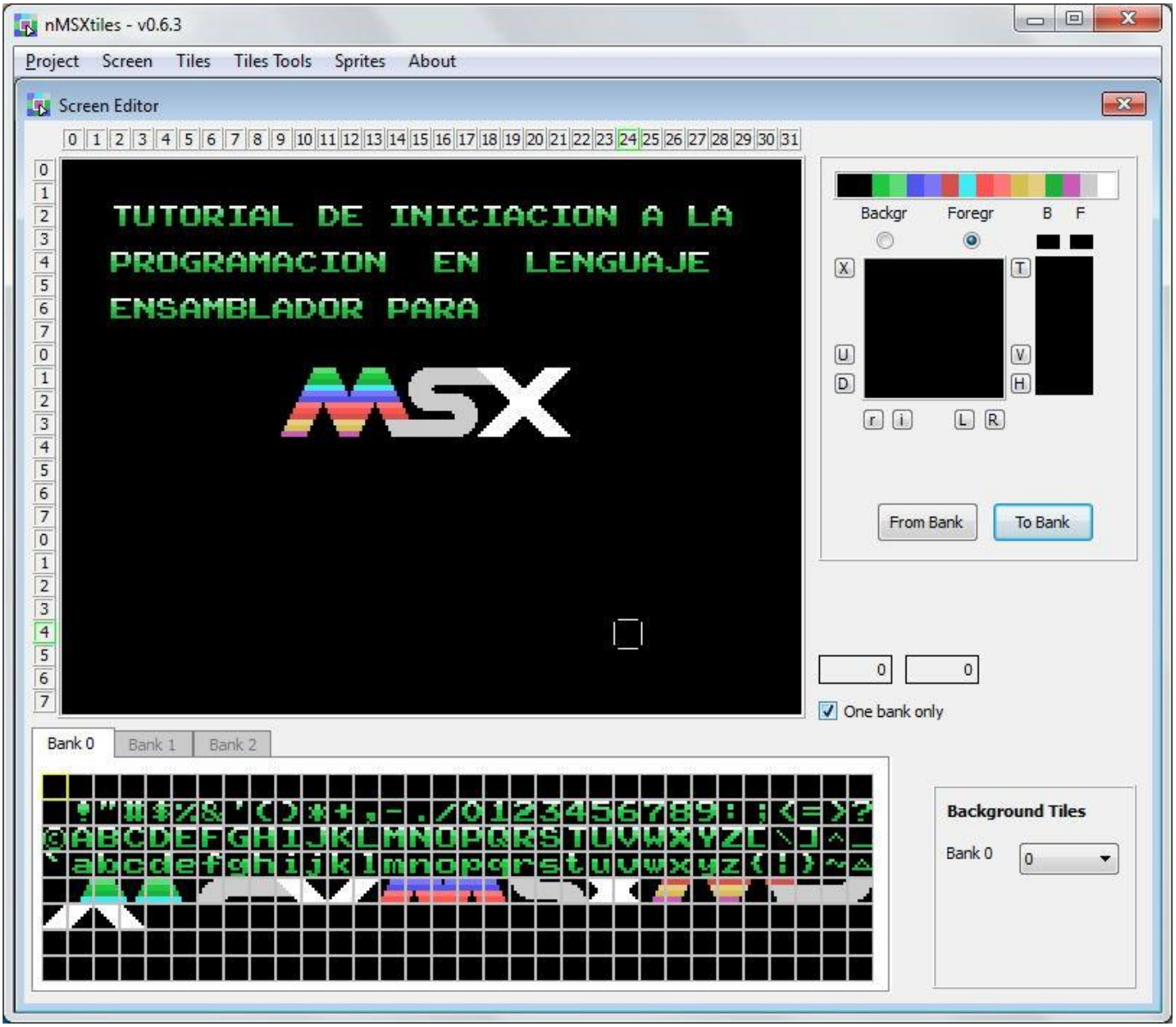
TUTORIEL D'INITIATION A LA PROGRAMATION EN LANGAGE ASSEMBLEUR POUR MSX

5eme PARTIE – LE MONDE DES GRAPHIQUES 3

Dans cette partie du tutoriel nous allons changer la façon de travailler afin de montrer le 'hola mundo grafico' d'une manière qui emploiera moins de code et d'octets, c'est la méthode que j'ai employé pour les menus de mon jeu JumpinG. Nous verrons comment importer nos images et graphiques dans nMSXtiles de Ramon de la Heras, avec une option que je lui ai demandé d'apporter. Nous verrons comment fonctionne ce programme que nous utiliserons pour tout le thème des menus et mappage. Ensuite nous finaliserons la partie du monde des graphiques.



Ceci est l'outil principal de ce tutoriel, le texte et le logo que nous obesrvons dans nMSXtiles sera le résultat final de notre tutoriel, que nous appelerons “HolaMundoGrafico4.asm” et qui est inclu dans le dossier Tutorial5.rar

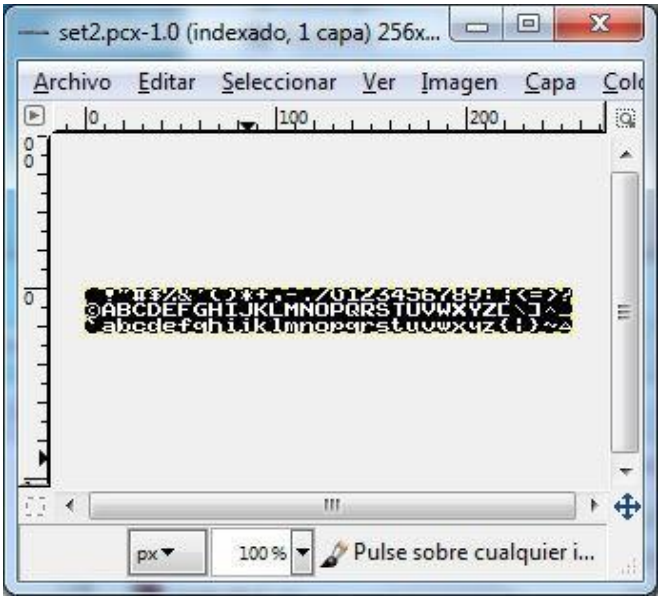
http://www.dimensionzgames.com/wp-content/uploads/downloads/2012/02/Tutorial5.rar

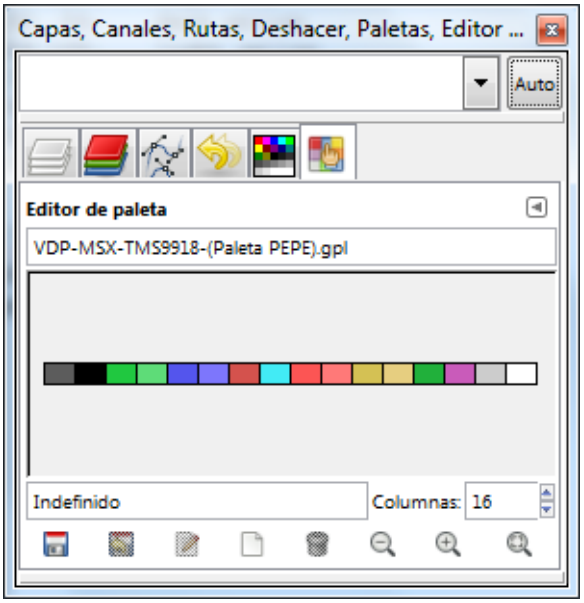
La première chose que l'on va faire c'est modifier la palette que nous utilsons sous GIMP, la couleur transparente et la couleur noire sont les mêmes et de ce fait, on ne peut différencier ces 2 couleurs. Nous devons entrer dans le répertoire où est installé GIMP, un dossier s'appelle palettes dans lequel il se trouve, ou bien faites une recherche sous Windows du fichier palette que nous utilisons VDP-MSX-TMS9918-(Paleta-PEPE).gpl , on peut utiliser EditPlus pour le modifier.

```
GIMP Palette
Name: VDP-MSX-TMS9918-(Paleta PEPE).gpl
Columns: 16
#
  0   0   0    Transparent
  0   0   0    black
 32 200  64    medium green
 94 220 120    light green
 84  85 237    dark blue
125 118 252    light blue
212  82  77    dark red
 66 235 245    cyan
252  85  84    medium red
255 121 120    light red
212 193  84    dark yellow
230 206 128    light yellow
 33 176  59    dark green
201  91 186    magenta
204 204 204    gray
255 255 255    Sin nombre
```

```
GIMP Palette
Name: VDP-MSX-TMS9918-(Paleta PEPE).gpl
Columns: 16
#
 92  92  92    Transparent
  0   0   0    black
 32 200  64    medium green
 94 220 120    light green
 84  85 237    dark blue
125 118 252    light blue
212  82  77    dark red
 66 235 245    cyan
252  85  84    medium red
255 121 120    light red
212 193  84    dark yellow
230 206 128    light yellow
 33 176  59    dark green
201  91 186    magenta
204 204 204    gray
255 255 255    Sin nombre
```

On peut voir que l'on a remplcé les trois 0 par trois 92 dans la couleur Transparent, de cette manière le noir sera noir et le transparent sera gris sombre. Sauvegardez ou enregistrez le fichier.
Maintenant nous allons ouvrir le jeu de caractères avec GIMP : fichier set2.pcx

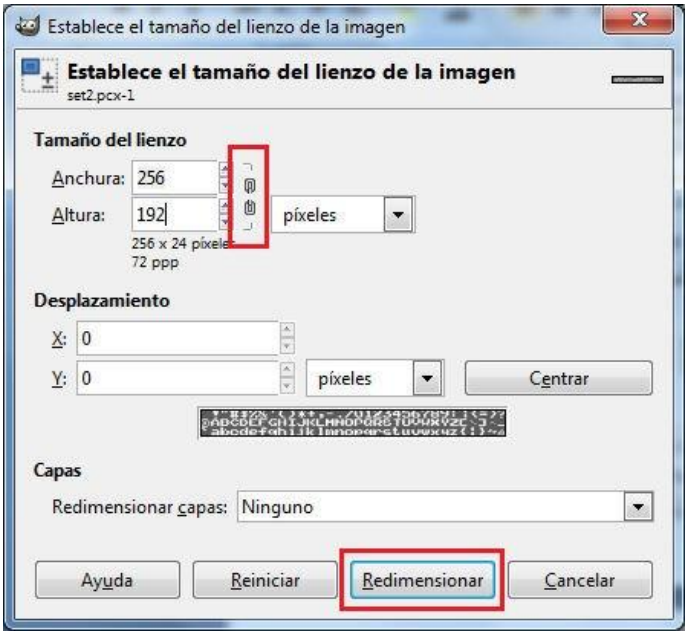




On peut voir l'image PCX avec notre jeu de caractères et la palette avec la modification de la couleur transparente qui apparait grise.

La première étape avant d'ouvrir une image est de changer la palette par défaut de GIMP par celle du MSX. Choisir dans les menus de GIMP : Couleurs-Mappage-Définir la palette des couleurs référez-vous à l'image à droite de ce texte pour le résultat.
Appuez sur Default et dans la fenêtre qui s'ouvre sélectionnez la palette VDP-MSX-TMS9918-(Paleta-PEPE). Appuyez sur Fermer.
Là où nous avions Default nous avons maintenant VDP-MSX-TMS9918-(Paleta-PEPE). Appuyez sur Accepter.

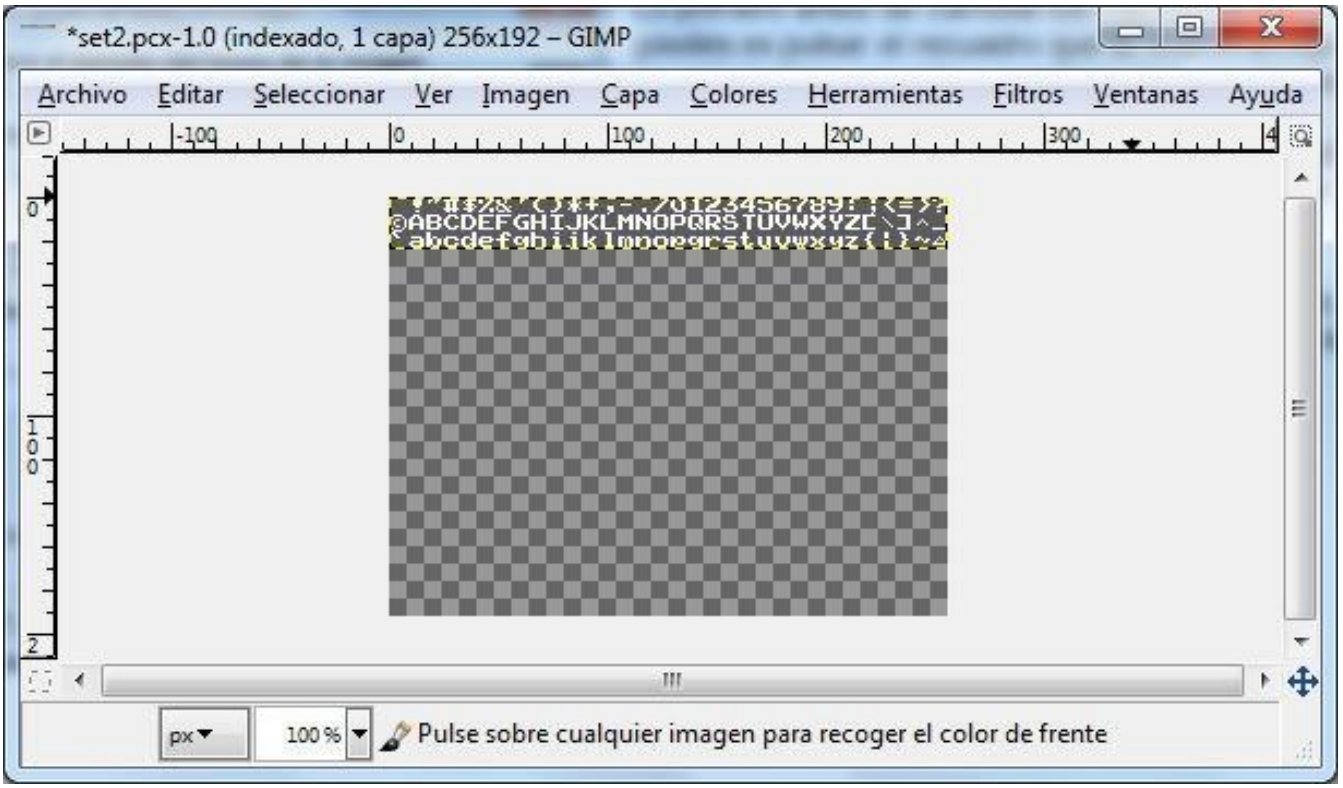
Nous allons maintenant modifier la dimension de l'image à 256x192 puisque l'importateur de fichier de nMSXtiles ne permet pas d'importer des résolutions plus réduites. Cela se fait via le menu de GIMP avec Image-Taille du canevas… (Cela n'altère pas l'image seulement la surface de travail).



Avant de changer les valeurs des pixels, appuyez sur la chaîne qui est encadrée en rouge sur l'image, cette option est pour créer une image proportionnelle et nous voulons plutôt créer une image aux dimensions définies.

Comme notre largeur est de 256, nous ne changeons rien mais passon la hauteur de 24 à 192. Maintenant appuyez sur Redimensionner. La fenêtre se ferme, assurez-vous que l'option 'redimensionner les calques' est positionné sur 'aucun'. Nous obtenons maintenant une image à la résolution de l'écran MSX 256x192 pixels.

Observez que nous avons modifié la résolution et gardé intact notre graphisme.

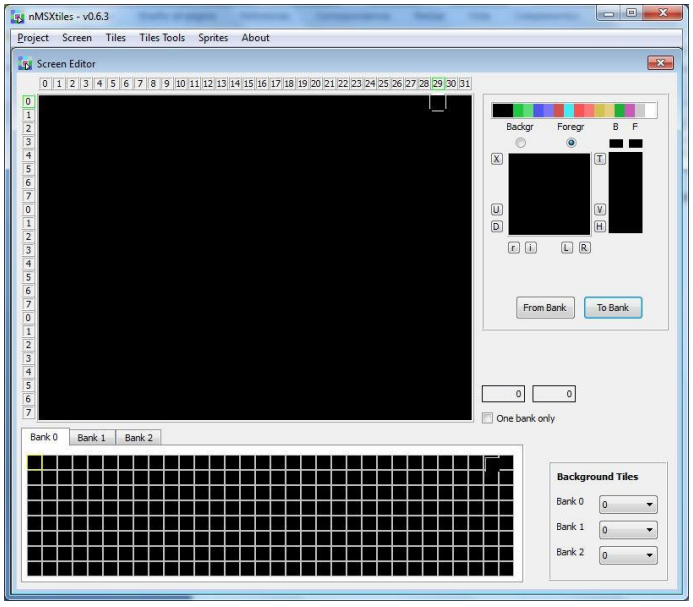


Nous allons maintenant enregistrer notre image au format PNG pour l'importer ensuite dans nMSXtiles. Appuyez dans GIMP sur Fichier , Enregistrer sous, dans la fenêtre qui s'ouvre, changez le nom set2.pcx par set2.png et enregistrez.

Nous avons un message d'avertissement car au changement de dimension nous avons eu un nouveau calque qui ne peut être récupéré en PNG. Choisissez le bouton Exporter et il se chargera de combiner les calques pour enregistrer le format PNG. Valider les formats d'exportation sans y toucher par Sauvegarder.

Nous sommes prêt à l'importer dans nMSXtiles, il nous faut seulement le fichier paletaMSX.png pour que nous puissions l'importer, dans le dossier TUTORIAL5.rar, prenez le fichier pour l'utiliser.
Ce PNG que nous avons créé contient un pixel de chaque couleur de la palette MSX dans l'ordre correct des numéros de couleurs, pour que nMSXtiles puisse les importer et les valeurs correspondantes à chaque partie de l'image.

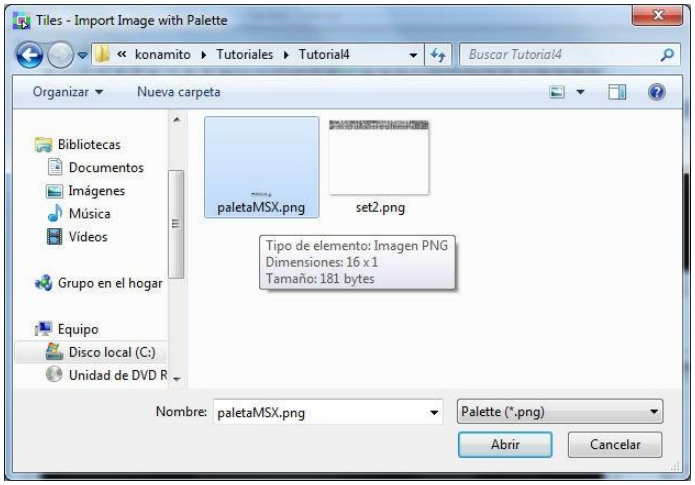
Décompressez le nMSXtiles 0.6.4 qui est dans le dossier TUTORIAL5.rar sous le nom nmsxtilesv0_6_4.zip dans votre dossier C:\MSX où se trouvent le reste des outils.
C'est une version beta qui est un peu instable mais c'est peu être dû à l'utilisation sous Windows 7 64 bits, mais je l'utilise car il est très utile pour extraire des blocs de caractères et autres.



Lancer nMSXtiles et nous allons créer un projet nouveau. Dans le menu appuyez sur Project – New et vous allez obtenir la même chose que sur l'image ci-contre.

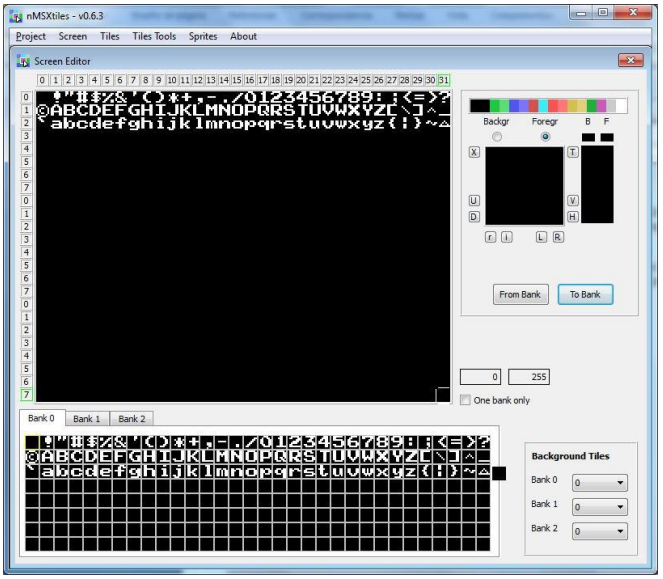
Nous allons maintenant importer l'image PNG du jeu de caractères créé avec GIMP. Appuyez sur Tiles - Import

On voit que la première chose que l'on va nous demander est le fichier avec la palette PNG, appuyez sur OK.



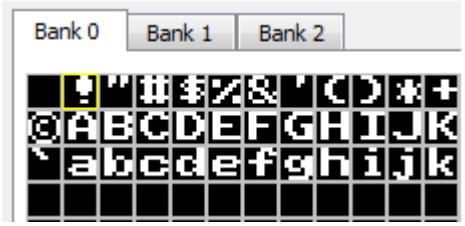
Sélectionnez le fichier paletaMSX.png qui se trouve dans les exemples de ce tutoriel. Et appuyez sur Ouvrir.

Ensuite, on nous demande le fichier d'image, sélectionnez le fichier set2.png et appuyez de nouveau sur Ouvrir.



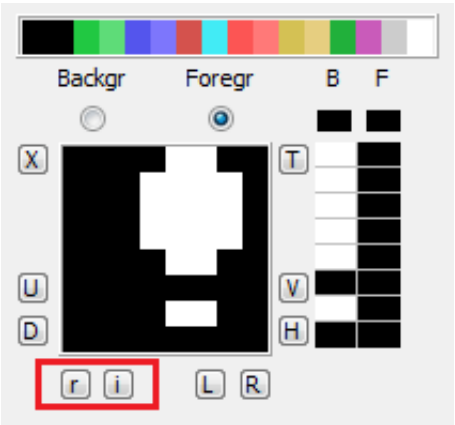
Et voilà… Nous avaons maintenant importé notre graphique créé avec GIMP ou avec le programme de graphisme qui vous convient comme photoshop, paint etc.

Nous allons voir maintenant d'autres aspects à prendre en compte au moment d'importer une image ou un graphique à notre projet dans nMSXtiles.

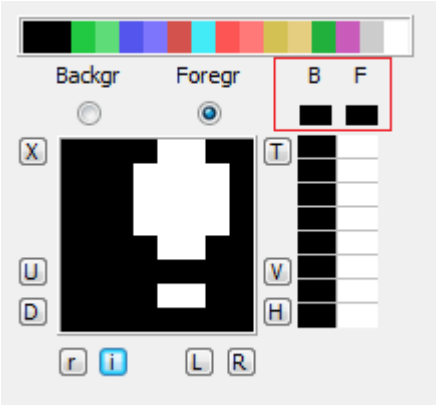


Dans la fenêtre des 'Bank' double-cliquez sur le symbole d'exclamation, comme encadré en jaune.

Je vous explique tout cela pour vos futures importations de de graphiques dans nmSXtiles en blanc et noir.

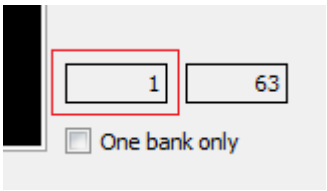


ici, dans la fenêtre d'édition du CHR on peut voir le symbole d'exclamation qui est de couleur blanche "B" pour le fond et la couleur noire pour les lettres "F" , les couleurs se sont inversées (qu'est-ce qui s'est passé ?), l'importateur a vu qu'il y avait 2 couleurs mais ne sait pas distinguer celle qui correspond au fond et celle des caractères. On peut le laisser tel quel mais il faudra en tenir compte au moment de colorer l'écran. Je le modifie en appuyant sur le petit bouton "i" pour inverser l'ordre des couleurs. Quand l'importation mélange les couleurs blanches et noires en “B” et “F” on peut appuyer sur “ r “ pour ré-ordonner toutes les couleurs d'un côté et optionnellement appuyer ensuite sur "i" pour les laisser en "B" ou "F". Ensuite on devra appuyer sur le bouton To Bank pour apporter la modification du caractère dans le Bank et l'appliquer à tous les autres.



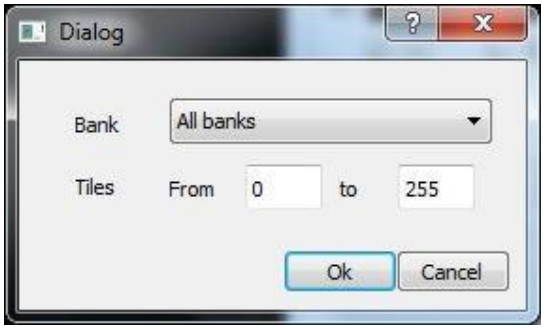
On peut aussi améliorer un autre aspect que l'importation ne prend pas en compte, si on regarde le point d'exclamation, il n'y a pas de couleur blanche sur la dernière ligne (octet) du CHR, ce qui est normal puisqu'il n'y a pas de pixel, il n'ya pas lieu d'en tenir compte. Pour qu'au moment de la compression on ne perde pas des octets blanc ou noir on peut appuyer dans la case noire en dessous du "F" qui est encadré en rouge et appuyez ensuite sur la couleur blanche , ainsi nous lui indiquons que tous les octets "B" ou "F" sont d'une même couleur. Bien que j'ai modifié de cette façon on pourrait mettre une couleur spécifique pour ces lettres, vous pouvez faire ces modifications vous-même.

Un autre truc, on peut modifier les couleurs d'un CHR puis copier ce CHR avec CONTROL+C puis se positionner sur le CHR suivant et avec CONTROL+G lui appliquer les mêmes couleurs pour gagner du temps en changeant les 8 octets d'un coup pour chaque CHR.



Nous allons voir une nouvelle option de cette version , extraire d'un fichier binaire un nombre de caractère défini, pour cela nous avons besoin de connaître le caractère de départ et de fin, cela on peut le voir via nMSXtiles en appuyant sur le CHR et dans l'image qui s'affiche à la droite de ce texte et qui est encadré en rouge. Le premier est le CHR 0 et le dernier est le CHR 95, avec ces données nous pouvons procéder et les extraire.

Appuyez dans le menu Tiles - Export bin data…

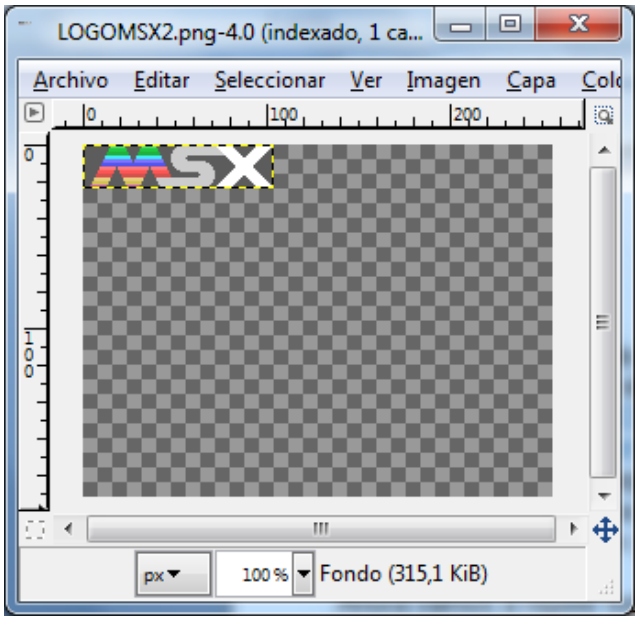


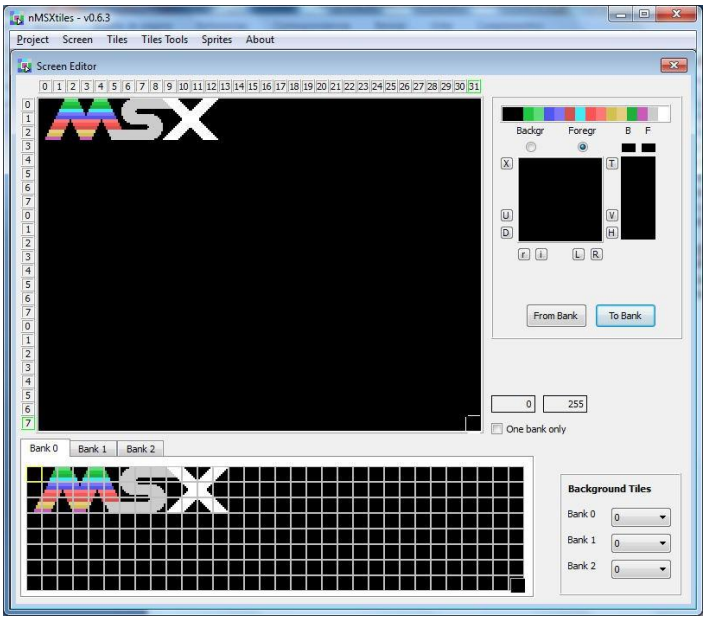
Dans la boite de dialogue qui s'ouvre mettre comme nom set2. Avant de valider cette fenêtre, il faut sélectionner le numéro de bank parmi les 3 et la plage de CHR que l'on veut extraire.

Ainsi, on sélectionne le Bank 0 et dans 'Tiles' on sélectionne 'From 0 to 95' et on valide par OK. Il a sauvegardé dans le répertoire 2 fichiers set2.til et set2.col qui sont les fichiers de CHR et CLR des CHR au format binaire. On peut compresser avec Pletter comme vu plutôt, et le convertir avec binDB.exe au format texte en ASM, bien qu'ici je n'en ai pas besoin.

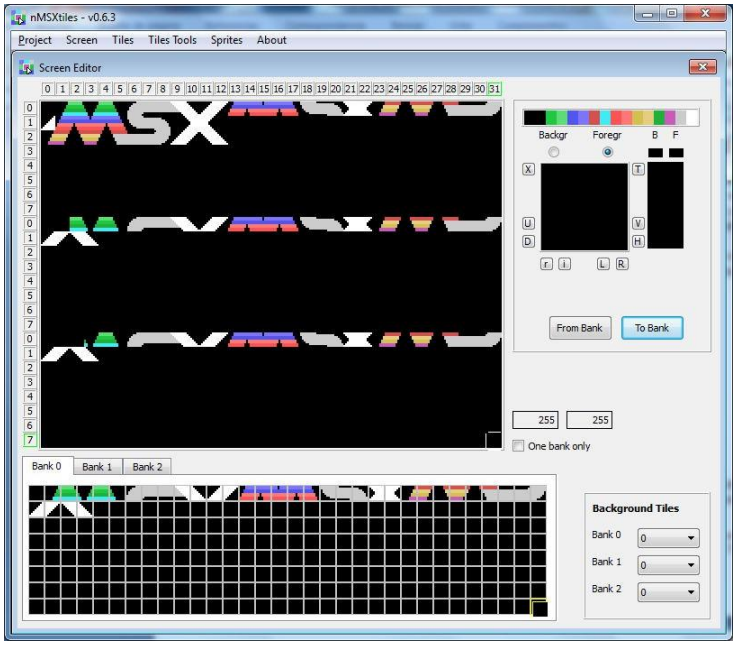
Continuons sur l'utilisation de nMSXtiles.
Maintenant, nous allons sauvegarder le projet qui se compose de 3 fichiers, un qui est le projet propre ".PRJ", l'autre le fichier de l'écran ".SCR" et le dernier fichier avec les 3 blocs de CHR et leurs couleurs ".TIL". Appuyez sur menu Project – Save , on nous demande d'abord le nom de l'écran, je vous donne un conseil, je crée un dossier avec le nom Proyecto_set2 et j'utilise le même nom pour les 3 fichiers, de telle manière qu'ils soient dans le même dossier. Comme j'ai spécifié les 3 fichiers avec set2 sans extensions, c'est le programme utilisé qui le fera, en l'enregistrant dans le classeur.

Nous allons répéter tous le processus de la page 2, pour importer le logo MSX dans nMSXtiles, comme il est ouvert appuyez sur Project - New et redimesionnez la fenêtre de travail, ensuite ouvrez GIMP avec le fichier LOGOMSX2.PCX et faites de même qu'expliqué à la page 2. Assurez-vous que la palette du MSX est bien sélectionnée, enregistrez au format PNG en applatissant les calques. Maximisez maintenant nMSXtiles afin qu'il prenne tous l'écran.





réalisez le même process vu en page 3, Menu Tiles - Import, sélectionnez le fichier avec la palette PNG, et ensuite l'image du logoMSX2.png, une fois obtenu ce que vous voyez sur la copie écran ci-dessus, assurez-vous que l'ordre des couleurs est correct.



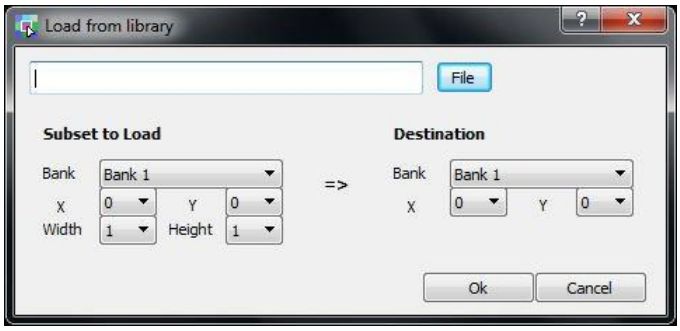
Nous allons ensuite juxtaposer les caractères car pour l'instant ce n'est pas le format requis pour exporter de tel numéro à tel numéro.

On peut le faire manuellement ou en utilisant une option du programme. Dans le menu Tiles Tools - Group Tiles on peut voir que les tiles se sont regoupées, nous allons maintenant sauvegarder le projet, créons un dossier Proyecto_logoMSX et sauvegardons les 3 fichiers sous le nom logoMSX.

Nous allons voir d'autres options de nMSXtiles, pour cela créons un nouveau projet.

Nous allons apprendre comment importer des tiles d'un autre fichier ou bloc , vous devez au prélable savoir quel fichier conserver et quels numéro de caractères vous voulez importer. Avec ceci nous allons générer unn bloc de CHR que nous utiliserons ensuite pour HolaMundoGrafico4.asm.

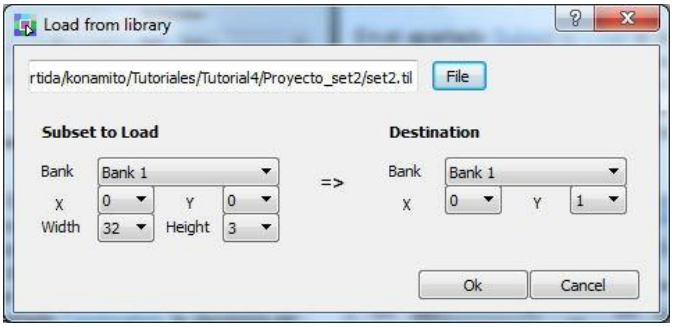
Appuyez sur le menu d'nMSXtiles Tiles – Load from library… la fenêtre suivante s'ouvre.



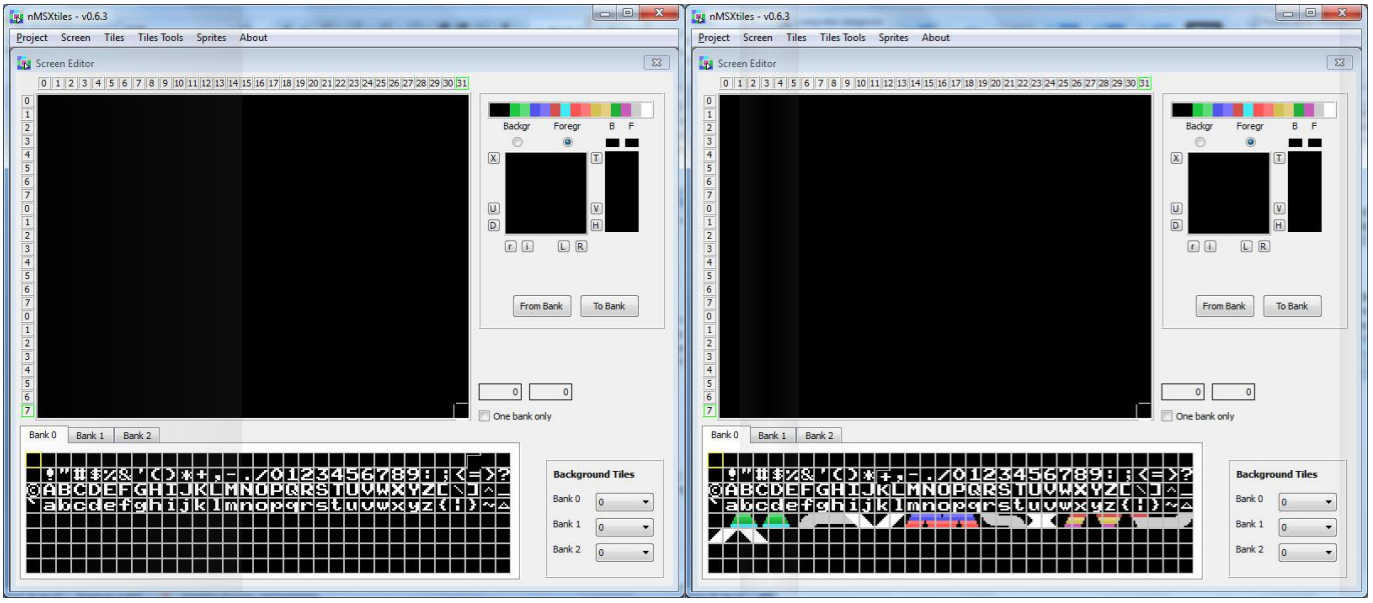
Appuyez sur File et sélectionnez le fichier set2.til qui contient les 3 blocs de caractères du répertoire Proyecto_set2

Dans le champs Subset to Load nous indiquons 3 informations.
1- lequel des 3 blocs de CHR nous allons lire,
2- à quel CHR débutons-nous en précisant le X et le Y des coordonnées du début de bloc.
3- quelle largeur et hauteur de CHR voulons-nous lire.

Nous voulons le bloc 1 “Bank 1”, “X=0 et Y=0” parce que nous voulons débuter au CHR0, la largeur est “Width 32” qui est la largeur du bloc et la hauteur “Height 3” qui représente les 3 lignes qu'occupent les lettres du bloc.
Dans le champs Destination nous indiquons dans quel bloc de notre projet actuel on veut récupérer les CHR. Ici j'ai choisi le bloc 1 “Bank 1” mais comme je souhaite que les caractères débutent à partire du CHR 32, je lui indique “X=0 e Y=1”

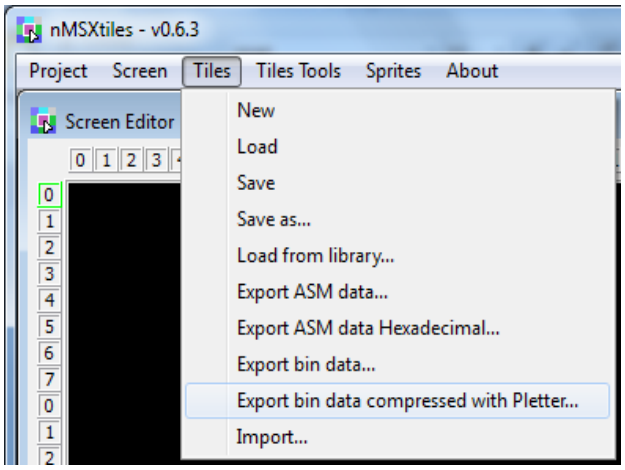


Le X et le Y sont la colonne et la ligne du bloc. Comme le CHR 32 débute à la colonne 0, ligne 1sachant que la colonne se compte de 0 à 31 et la ligne de 0 à 7 pour chaque bloc. Ainsi on met X=0 et Y=1.
Appuyez sur OK. Vous devriez obtenir l'image ci-dessous à gauche.

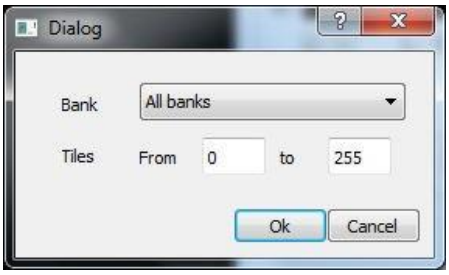


On va maintenant répéter le même process pour importer les caractères du logo MSX, je pense que vous savez déjà le faire, une explication rapide, menu Tiles-Load from library lire le fichier logoMSX.til du répertoire Proyecto_logoMSX dans le champs Subset to Load mettre “Bank 1” et “X = 0 e Y = 0” avec “Width 32” et “Height 2”,dans le champs Destination mettre “Bank 1” et “X = 0 e Y = 4” et validez par OK.
Le résultat doit ressembler à l'image ci-dessus à droite. Nous avons notre bloc 0 prêt pour notre exemple HolaMundoGrafico4.asm que nous allons voir ensuite, mais avant voyons une denière chose sur nMSXtiles.

Le Bank 0 est prêt mais si on appuit sur le bank 1 ou 2 on voit que ce n'est pas le cas, or nous avons besoin d'avoir les mêmes blocs alors que l'on pensait que la modification du bloc 0 se reportait automatiquement sur les autrtes blocs. Mais c'est plus facile que vous ne le pensez, nMSXtiles à une checkbox appelée One bank only juste en dessous d'où on peut voir le numéro du caractère. Si on marque ou démarque cette checkbox cela copie le bloc 0 sur le bloc 1 et le bloc 2. De cette manière nous obtenons 3 blocs identiques, vous pouvez sélectionner les blocs pour le vérifier, mais si on veut créer des menu c'est plus simple de marquer cette checkbox pour avoir les mêmes blocs. L'étape suivante est de sauvegarder les 3 fichiers du projet avec le nom tutorial5 dans un nouveau répertoire appelé Proyecto_tutorial5.



Autre chose important et commode c'est que l'on peut directement exporter les blocs de CHR au format binaire et compressé avec Pletter sous nMSXtiles, allons dans le menu Tiles - Export bin data compressed with Pletter...
Sauvegarder avec le nom banco0.plet

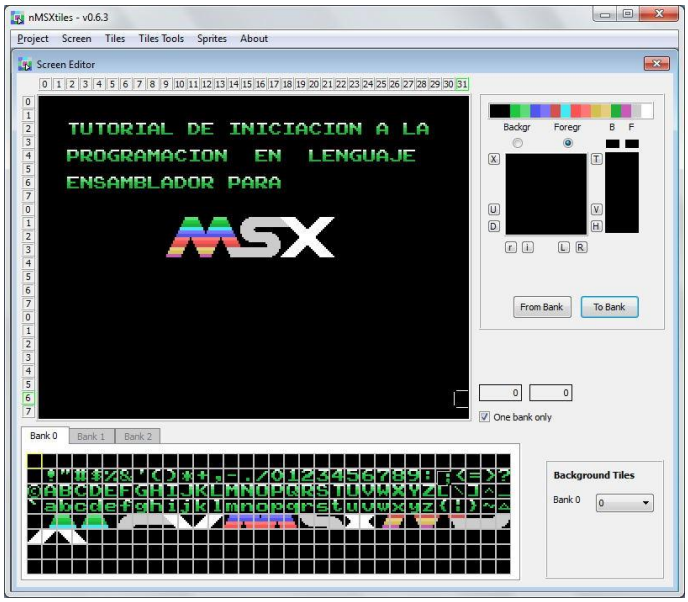


Sélectionnez Bank 0

et Tiles From 0 to 255

Et bouton OK

Maintenant nous avons créé 2 fichiers compressés avec Pletter pour les CHR et CLR, les fichiers se nomment banco0.plet.til et banco0.plet.col que nous allons utiliser dans notre code.

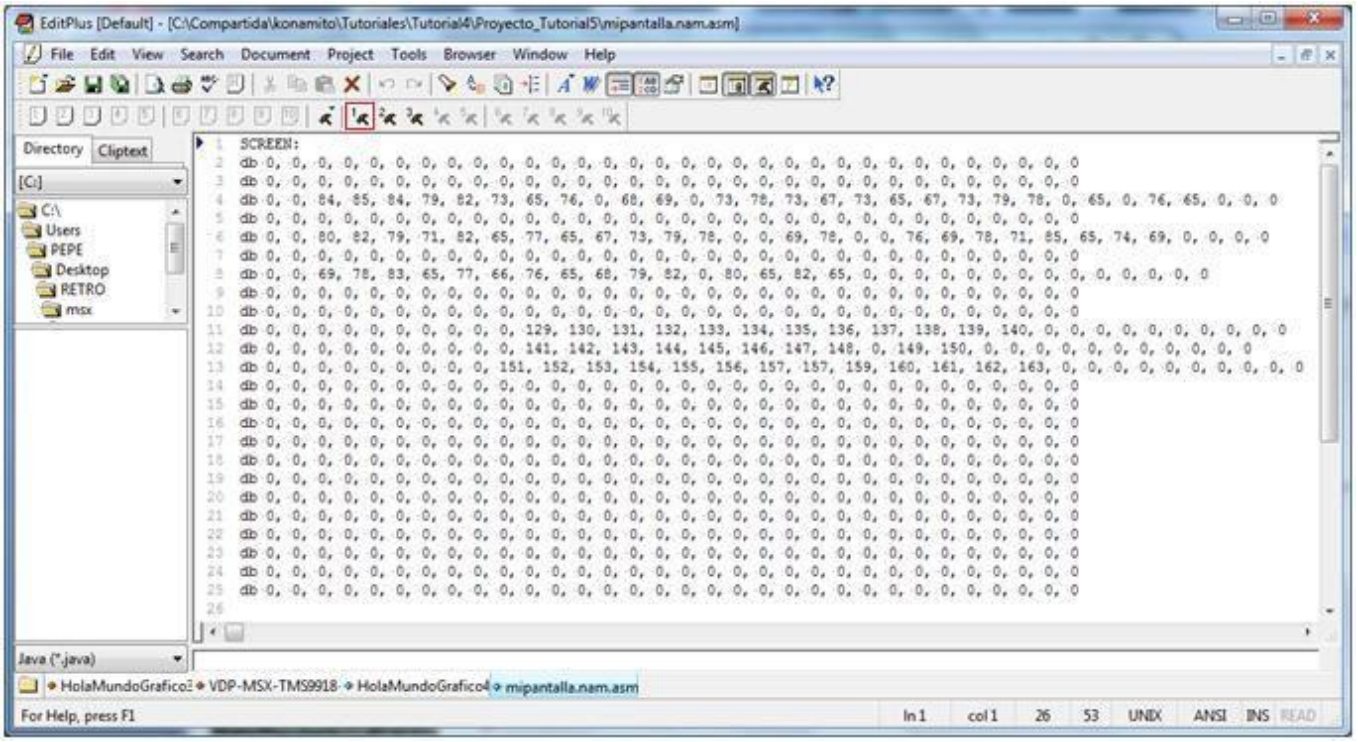


Nous pouvons appeler notre texte et le logo MSX ou bien comme dans l'image ci-contre, construire librement l'écran sous nMSXtiles. Il faut sélectionner les caractères désirés du bloc de CHR et cliquer sur la zone de l'écran pour le mettre à la position souhaitée.

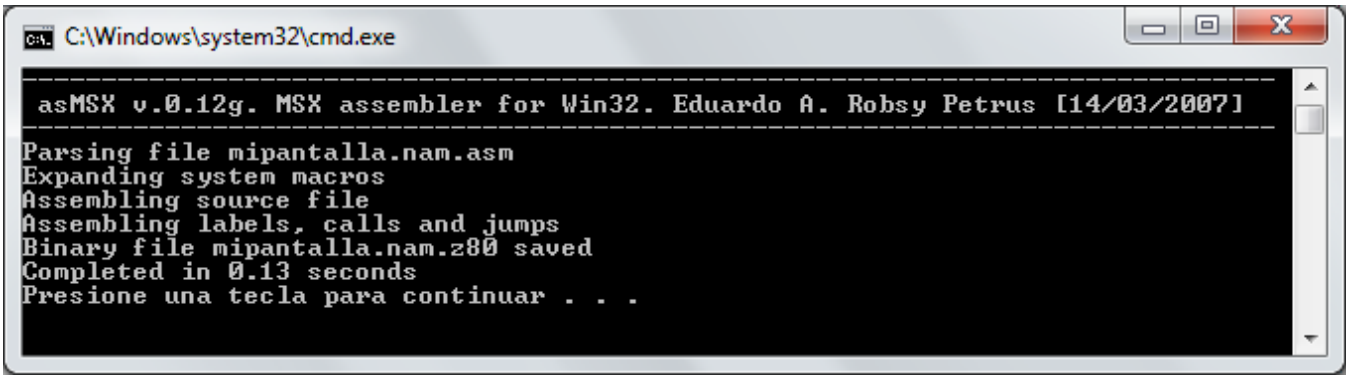
Une fois l'écran créé nous allons la sauvegarder et compressée au format Pletter, pour cela appuyez sur le menu Screen – Export bin data compressend with Pletter... dans la boîte de dialogue choisissez le répertoire du projet du tutorial5 et donnez-lui le nom mipantalla.nam.plet de cette manière on repèrera que notre écran est le ".nam" qui sera pour la NAMTBL et ".plet" pour indiquer que c'est compressé avec Pletter.

Dans mon cas l'écran l'écran occupe 768 Octets mais compressé avec Pletter il ne fait plus que 129 Octets, moins que tous le code qu'il nous faudrait pour créer le texte et graphisme. Nous verrons le nouveau code quand nous allons coder le HolaMundoGrafico4.asm

Nous avons maintenant tout ce qu'il nous faut. Mais je vais vous expliquer une chose qui sera utile pour la suite. De même que l'on peut sauvegarder l'écran en binaire ou en binaire compressé avec Pletter, on sauvegarder l'écran au format ASM en DB, cela peut-être utile si on veut le rattacher au code, ou bien quand on ne veut pas utiliser tout l'écran parce que nous avons besoind'un tiers ou deux tiers de la zone pour un autre usage. Pour sauvegarder au format DB, allez dans le menu Screen – Export ASM data… et dans la boîte de dialogue mettre mipantalla.nam.asm et sauvegarder dans le répertoire du projet. Regarder dans ce répertoire et ouvrez avec EditPlus le fichier, vous devriez obtenir la même chose que la copie écran de la page suivante, on on peut voir les 768 Octets composés de 32 CHR de large x 24 CHR de haut.



Il est ainsi facile d'éliminer certaines lignes du code car c'est du teexte éditable. Je l'ai utilisé pour mon jeu JumpinG afin d'enlever les 2 premières lignes car je les utilisais pour les scores, mais revenons à la suppression des lignes et du label SCREEN: qui ne nous intéresse pas, pour passer en binaire, il suffit de compiler avec EditPlus.



On peut voir ici que l'assembleur asMSX a généré un fichier binaire appelé mipantalla.nam.z80 que l'on peut compresser avec Pletter et ajouter à votre code via un .incbin “NomDuFichier”

Fermez nMSXtiles et continuons avec EditPlus 3 , créez un nouveau document et tapez le code suivant.

```
;-----------------------------------------------------
; Nom de notre programme
; Hola Mundo Grafico 02/02/2012
; Version 4
;-----------------------------------------------------

;-----------------------------------------------------
; CONTANTES
;-----------------------------------------------------
; Nous n'avons pas défini de constantes

;-----------------------------------------------------
; VARIABLES DU SYSTEME
;-----------------------------------------------------
; Adresses de la VRAM
        CHRTBL          equ     0000h   ; Table des caractères
        NAMTBL          equ     1800h   ; Table des Noms
        CLRTBL          equ     2000h   ; Table des couleurs des caractères
        SPRATR          equ     1B00h   ; Table des attributs de SPRITES
        SPRTBL          equ     3800h   ; Table des SPRITES
```

```
; Variables du système MSX
       CLIKSW          equ     $F3DB   ; Keyboard click sound
       FORCLR          equ     $F3E9   ; Foreground colour

;-------------------------------------------------------
; DIRECTIVES POUR L'ASSEMBLEUR ( asMSX )
;-------------------------------------------------------
       .bios           ; Definir les noms des appels du BIOS
       .page 2         ; Definir l'adresse du code assembleur 8000h
       .rom            ; On indique que l'on créé une ROM
       .start INICIO   ; Début du code de notre programme
; Suivre la norme du Standard MSX pour une ROM
       dw 0,0,0,0,0,0 ; 12 octets à 0

;-------------------------------------------------------
; DEBUT DU PROGRAMME
;-------------------------------------------------------
INICIO:
       call    INIT MODE SCx          ; activer le mode graphique x
       call    INIT_GRAFICOS          ; charger les graphismes en VRAM

; Charger la NAMTBL compressée en VRAM
       ld      hl,MIPANTALLA          ; source des octets pour la NAMTBL
       ld      de,NAMTBL              ; destination  NAMTBL
       call    DEPLET                 ; décompresser en VRAM

FIN:
       jp      FIN            ; c'est comme 100 goto 100 pour faire une boucle sans fin.

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

;-------------------------------------------------------
; INITIALISE LE MODE ECRAN ET LES COULEURS
;-------------------------------------------------------
; BASIC: COLOR 15,1,1
; Chargement des couleurs
INIT_MODE_SCx:
       ld      hl,FORCLR      ; Variable du système
       ld      [hl],15        ; Couleur du premier plan 15=blanc
       inc     hl             ; FORCLR+1
       ld      [hl],1         ; Couleur de fond 1=noir
       inc     hl             ; FORCLR+2
       ld      [hl],1         ; Couleur du bord 1=noir
; call INITXT          ; BIOS set SCREEN 0
; call INIT32          ; BIOS set SCREEN 1
       call    INIGRP         ; BIOS set SCREEN 2
; call INIMLT          ; BIOS set SCREEN 3
;
; SCREEN 0 : texte de 40 x 24 sur 2 couleurs
; SCREEN 1 : texte de 32 x 24 sur 16 couleurs
; SCREEN 2 : graphiques de 256 x 192 pixels sur 16 couleurs
; SCREEN 3 : graphiques de 64 x 48 pixels sur 16 couleurs

; Ceci enlève le son émis par les touches du msx quand on appuit sur le clavier
       xor     a                      ; ld a,0
       ld      [CLIKSW],a             ; Variable BIOS désactiver le son des touches
; quitter la routine INIT_MODE_SCx
       ret
;-------------------------------------------------------

;-------------------------------------------------------
; CHARGER LES GRAPHISMES EN VRAM
;-------------------------------------------------------
INIT_GRAFICOS:
; On effectue ceci afin que l'on ne voit rien à l'écran
; pendant que l'on charge les CHR et les couleurs en VRAM
       call    DISSCR                 ; BIOS inhibe l'écran

; Remplir les 768 Octets de la Name Table - NAMTBL en VRAM
       ld      hl,NAMTBL              ; Adresse de début dans la VRAM
       ld      bc,768                 ; nombre d'octets à remplir 32 colonnes * 24 lines=768
       xor     a                      ; a=0 – valeur à utiliser pour remplir
       call    FILVRM                 ; BIOS -Fill block of VRAM with data byte

; nous allons charger le bloc de CHR dans les 3 blocs de la CHRTBL en VRAM
     ; tiers 1
       ld      hl,banco0 CHR          ; source des octets des CHRs
       ld      de,CHRTBL              ; destination CHRTBL tiers 1
       call    DEPLET                 ; décompresser en VRAM
```

```
        ; tiers 2
                ld      hl,banco0 CHR           ; source des octets des CHRs
                ld      de,CHRTBL+2048          ; destination CHRTBL tiers 2
                call    DEPLET                  ; décompresser en VRAM

        ; tiers 3
                ld      hl,banco0_CHR             ; source des octets des CHRs
                ld      de,CHRTBL+4096            ; destination CHRTBL tiers 3
                call    DEPLET                    ; décompresser en VRAM

; on va charger les blocs de CLR dans les 3 blocs de la CLRTBL en VRAM
        ; tiers 1
                ld      hl,banco0 CLR           ; source des octets des CLRs
                ld      de,CLRTBL               ; destination CLRTBL tiers 1
                call    DEPLET                  ; décompresser en VRAM

        ; tiers 2
                ld      hl,banco0 CLR           ; source des octets des CLRs
                ld      de,CLRTBL+2048          ; destination CLRTBL tiers 2
                call    DEPLET                  ; décompresser en VRAM

        ; tiers 3
                ld      hl,banco0 CLR           ; source des octets des CLRs
                ld      de,CLRTBL+4096          ; destination CLRTBL tiers 3
                call    DEPLET                  ; décompresser en VRAM

; On active à nouveau l'écran.
         call   ENASCR                  ; BIOS habilitar la pantalla

; Quitte la routine INIT_GRAFICOS
        ret
;--------------------------------------------------------

;--------------------------------------------------------
; Routine de décompression RAM/ROM en VRAM pletter v1.1
; XL2S Entertainment
;--------------------------------------------------------
        .INCLUDE        "DEPLET.asm"
;--------------------------------------------------------

;--------------------------------------------------------
; Bloc de CHRs compressé avec Pletter
;--------------------------------------------------------
banco0_CHR:
        .INCBIN         "Proyecto_Tutorial5\banco0.plet.col"
;--------------------------------------------

;--------------------------------------------------------
; Bloc de CLRs compressé avec Pletter
;--------------------------------------------------------
banco0_CLR:
        .INCBIN         " Proyecto_Tutorial5\banco0.plet.col"
;--------------------------------------------

;--------------------------------------------------------
; mon écran graphique pour la NAMTBL compressé avec Pletter
;--------------------------------------------------------
MIPANTALLA:
        .INCBIN         " Proyecto_Tutorial5\mipantalla.nam.plet"
;--------------------------------------------


;-------------------------------------------
; FINAL DE NOTRE CODE EN ASSEMBLEUR.
;-------------------------------------------
```

A ce niveau je suis sûr que vous savez ce que va faire ce code HolaMundoGrafico4.asm, mais je vais vous expliquer certains passages quand même.

L'entête des variables du programme et les adressages en VRAM sont plus complets que dans les précédents tutoriels, je pense que la partie la moins commentée concerne la norme qui suit le .start INICIO , qui sert à être compatible avec la norme d'une ROM, il faut 12 octets à 0 avant le début de notre code, cela devrait fonctionner sans mais il est toujours bon de suivre la norme du standard.

La routine INIT_MODE_SCx comme toujours se positionne en SCREEN2, on lui ajoute la suppression du son des touches. On verra plus tard que l'on peut compléter cette routine.

La routine INIT_GRAFICOS est beaucoup plus fournie mais c'est compréhensible car on désactive l'écran, pour vider l'écran on remplit la NAMTBL avec les CHR et les CLR, ensuite on décompresse les octets du bloc 0 que l'on a créé sous nMSXtiles dans la VRAM, dans les 3 tiers de l'écran, on répète l'opération pour les couleurs des CHR, pour finalement ré-activer l'écran et voir le résultat.

Au lieu de charger dans la NAMTBL à partir d'une colonne et d'une ligne différents textes appelés depuis une table pour chaque ligne, nous passons plutôt par une décompression de l'écran dans la NAMTBL nous évitant tout ce boulôt.

```
; Charger dans la NAMTBL compressé en VRAM
        ld      hl,MIPANTALLA           ; source des octets pour la NAMTBL
        ld      de,NAMTBL               ; destination  NAMTBL
        call    DEPLET                  ; décompresser en VRAM
```

Nous allons étudier la mémoire utilisée car c'est très important en langage machine. On va compter les octets qui composent le tutoriel4 et le comparer au résultat de ce tutoriel.

Format du tutorial4 :
Set de caractères CHRs = 504 octets, LogoMSX CHRs =153 octets, LogoMSX CLRs = 144 octets – TOTAL 801 octets
Nouveau Format :
Set de caractères CHRs + LogoMSX CHRs = 623 octets- set de caractères CLRs + LogoMSX CLRs = 181 octets – TOTAL 804 octets

On peut observer que l'on a 3 octets de plus mais nous sommes passé en mode multicouleur pour les lettres.

Format du tutorial4:
Code 86 octets+ texto 18 octets+ table des couleurs 8 octets+ table du logo 39 octets+ routine copyblock 20 octets = TOTAL 171 octets
Nouveau Format : (comme dans le tutorial4, l'écran est compressé avec le texte HOLA MUNDO Grafico et le logo MSX en 1 tiers) Code 27 octets + Ecran HolaMundo.nam.plet = 81 octets – TOTAL 108 octets

Ainsi le format du tutorial4 utilise au total 972 octets, alors que celui de ce tutoriel, avec le même résultat, 912 octets, nous avons gagné 60 octets, et en plus on s'est facilité la tâche tout en réduisant le code.
Continuons avec l'exemple :

```
;-----------------------------------------------------------
; Routine de décompression RAM/ROM en VRAM pletter v1.1
; XL2S Entertainment
;-----------------------------------------------------------
        .INCLUDE        "DEPLET.asm"
;-----------------------------------------------------------
```

J'ai ici supprimé le reste du code de la routine de décompression, je l'ai sauvegardé dans un fichier avec le nom DEPLET.asm. Avec .INCLUDE “nom_du_fichier” j'ain intégré dans le code en assembleur au même endroit l'adressage vers la routine, c'est le même effet que si on collait la routine à cet endroit le code la routine qui est dans le fichier DEPLET.asm , de cette manière on peut séparer des routines ou parties de code que nous savons qu'elles fonctionnent, et don’t nous n'avons pas besoin d'avoir tout le code sous les yeux car il prend beaucoup d'espace. Ou bien pour pouvoir changer plus-tard de routine de décompression si on utilise un autre programme de décompression.

```
;-----------------------------------------------------------
; Bloc de CHRs compressé avec Pletter
;-----------------------------------------------------------
banco0_CHR:
        .INCBIN         "Proyecto_Tutorial5\banco0.plet.til"
;-----------------------------------------

;-----------------------------------------------------------
; Bloc de CLRs compressé avec Pletter
;-----------------------------------------------------------
banco0_CLR:
        .INCBIN         "Proyecto_Tutorial5\banco0.plet.col"
;-----------------------------------------
```

Ansi on peut substituer au code des octets graphiques une directive qui va pointer vers le fichier qui les contiendra .INCBIN “nom_du_fichier” , pour intégrer un fichier binaire dans cette partie du code, j'utilise la directive .include mais au lieu d'ajouter l'extension .asm , j'ajoute des DB avec des données, mais avant la directive je crée un label banco0_CHR: ou banco0_CLR: pour que le code puisse savoir à quelle adresse débute les données et ce que l'on va en faire.
Avant de mettre à jour tout cela, nous devons passer le fichier dans le programme binDB et donner le nom du fichier dans le code, on peut voir ci-dessous que j'ai aussi donné la direction du répertoire où se trouve le fichier Proyecto_Tutorial5

```
;-------------------------------------------------------
; mon écran graphique pour la NAMTBL compressé avec Pletter
;-------------------------------------------------------
MIPANTALLA:
        .INCBIN         " Proyecto_Tutorial5\mipantalla.nam.plet"
;-----------------------------------------
```

Et pour finir, j'ai ajouté le label MIPANTALLA: pour donner la bonne direction au code pour trouver et utiliser mon fichier écran binaire compressé avec Pletter.
De cette manière, je peux indiquer dans le code les labels qui se chargerons de chercher les données dans un fichier annexe, ici par exemple MIPANTALLA que je vais décompresser dans la NAMTBL.

```
; Charger dans la NAMTBL compressé en VRAM
        ld      hl,MIPANTALLA           ; source des octets pour la NAMTBL
        ld      de,NAMTBL               ; destination  NAMTBL
        call    DEPLET                  ; décompresser en VRAM
```

Autre information , nous devons indiquer le nombre d'octet à transférer pour que la routine de décompression Pletter que j'ai implémenté dans le code fonctionne et sache quand terminer la décompression des données.

Le résultat final sera cette image à gauche.

Nous expliquerons dans le prochain tutoriel comment manipuler les SPRITE et comment les importer via notre code dans le programme. On fera aussi une revue des outils dédiés.

José Vila Cuadrillero

"ES DETESTABLE ESA AVARICIA ESPIRITUAL QUE TIENEN, LOS QUE SABIENDO ALGO, NO PROCURAN LA TRANSMISION DE ESOS CONOCIMIENTOS."
Miguel de Unamuno
Escritor y Filósofo.
(Bilbao 1864 - Salamanca 1936)