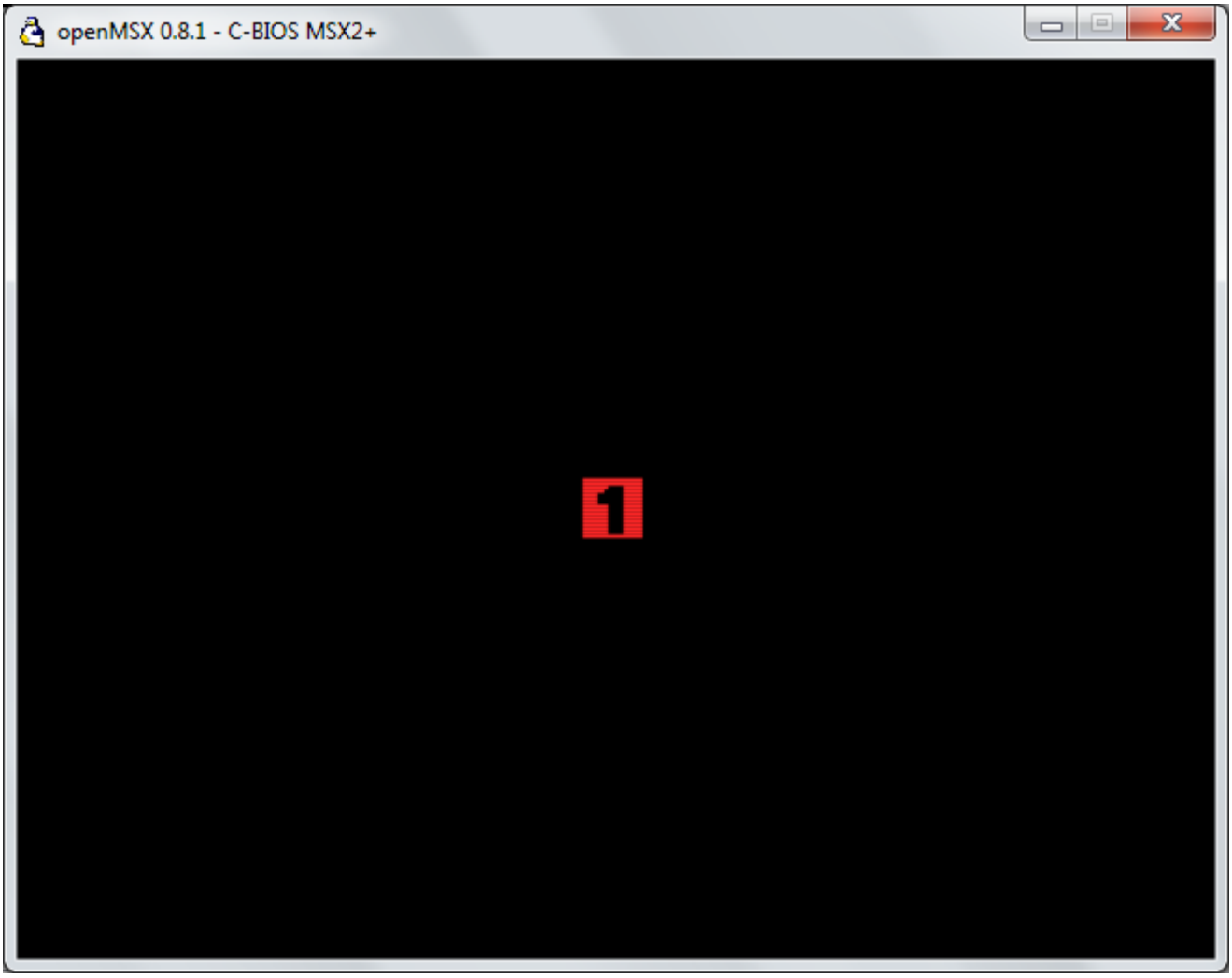
TUTORIAL DE INICIACION A LA PROGRAMACION EN LENGUAJE ENSAMBLADOR PARA MSX

6ª PARTE – EL MUNDO DE LOS SPRITES 1

En esta parte del tutorial veremos cómo funciona el mundo de los sprites en el MSX1 crearemos varios ejemplos de código, empezando con un Hola Sprite que iremos modificando, veremos la parte de la teoría, así como las herramientas que necesitaremos para trabajar con sprites.



Antes de empezar con el código del HolaSPR1.asm, vamos con la parte que menos os gusta, la teoría… ya sabéis que siempre es necesario para comprender los que queremos realizar.

Recordáis que en la tercera entrega en la pagina 4 cuando os explicaba las posiciones de la memoria VRAM deje unas direcciones sin explicar. (Os reproduzco el fragmento.)

> Queda la Sprite Pattern Table - SPRTBL y la Sprite Attribute Table - SPRATR que veremos más adelante en otra entrega del tutorial dedicada al mundo de los Sprites.

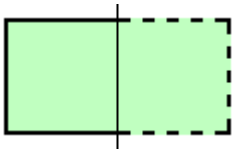
Pues ha llegado el momento de explicarlo. La Sprite Pattern Table - SPRTBL abreviado, es la zona de la VRAM comprendida entre las posiciones de memoria 3800h a 3FFFh ocupa 2Kb y es donde colocamos los CHRs de nuestros SPRITES recordar que en el MSX todo funciona con CHRs de 8x8 pixeles, como tenemos 2048Kb para sprites cuantos CHRs son?, fácil (2048 / 8) = 256 CHRs como máximo numerados del 0 al 255, podremos almacenar en esta parte de la memoria 256 Sprites de 8x8 o 64 sprites de 16x16 pixeles, pero el hardware del MSX solo nos permite usar 32 Sprites* simultáneos en pantalla al mismo tiempo. Y no puedo poner más de 64 sprites en VRAM? siempre puedes volcar a esta zona de la VRAM más SPRs a medida que los vayas necesitando o cuando se cambia de fase.

El área que maneja los sprites en la pantalla se llama Sprite Attribute Table - SPRATR abreviado, es la zona de la VRAM comprendida entre las posiciones de memoria 1B00h a 1B7Fh ocupa 128 Bytes aquí colocamos los Atributos de los Sprites que hacen que estos se visualicen en la pantalla. Permitirme abreviar estos términos de la siguiente manera ATRs son los Atributos de los Sprites y SPRs es como abrevio a los Sprites. Esta zona de la VRAM de 128 Bytes está compuesta por una tabla de 4 valores. Si dividimos 128 entre 4 (128 / 4) = 32 Que son el número máximo de sprites que podemos usar al mismo tiempo en la pantalla. Como se compone esta tabla te la explico a continuación

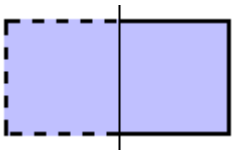
1º Valor – Coordenada Y - Esta es la coordenada Y del SPR su valor va desde 0 hasta 255.
2º Valor – Coordenada X - Esta es la coordenada X del SPR su valor va desde 0 hasta 255
3º Valor – Numero de CHR – que CHR dentro de la SPRTBL usaremos para el SPR de 0 a 255
4º Valor – Numero de Color – Aquí le decimos 2 cosas: 1-Que color de 0 a 15 de la paleta del MSX.

La segunda cosa en el Cuarto valor se conoce como EC o Early Clock si activamos el bit 7 de este byte lo que hace es restar 32 pixeles a la coordenada X del SPR, (Sigo sin entender esto)

Si os fijáis en muchos juegos cuando el personaje sale por el lado derecho de la pantalla desaparece poco a poco o pixel a pixel (No en todos) Sin embargo cuando lo hacen por el lado Izquierdo no sucede esto en la mayoría de juegos.



Esto sucede porque cuando el SPR llega a la coordenada X 0 esta pasa automáticamente a la 255 apareciendo el SPR por el lado derecho de la pantalla, para que esto no suceda, si activamos el EC, le resta 32 pixeles a la coordenada X del SPR.



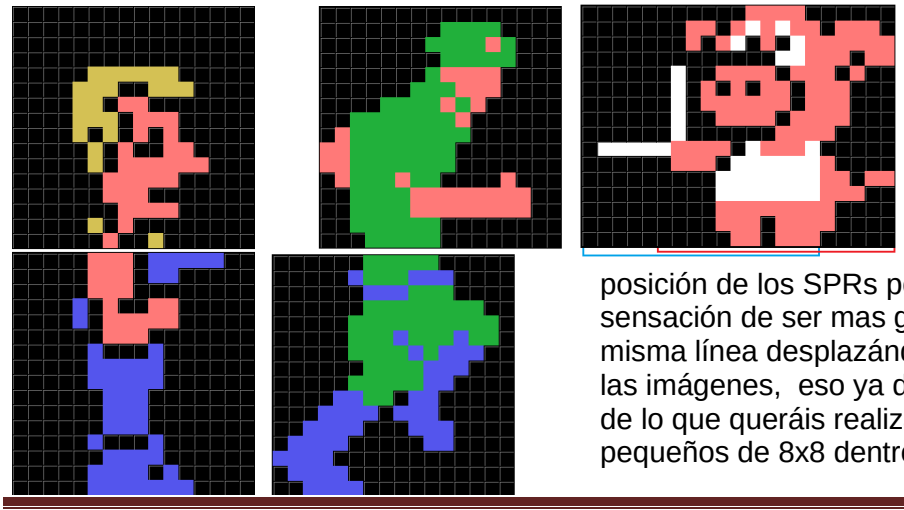
De esta manera puedes seguir restando la coordenada X para que se produzca una salida por el lado izquierdo igual que lo hace por el lado derecho, pero tenemos que estar gestionando por software el control de esta opción que suele ser muy engorrosa por eso en casi todos los juegos omiten hacerlo. Yo incluido.

Hay cuatro formas de definir los SPRs: 2 normales y 2 agrandados o ampliados.

1 – Sprites de 8x8 normales
2 – Sprites de 8x8 agrandados
3 – Sprites de 16x16 normales
4 – Sprites de 16x16 agrandados

Si activamos la opción del VDP para agrandar los sprites lo que hace es doblar el número de pixeles el tamaño no cambia sino que se amplía un SPR de 16x16 pasa a ocupar en la pantalla 32x32 y uno de 8x8 pasa a ocupar 16x16, pero no cambiamos los CHRs de los sprites sino que se hace un zoom.

Solo podemos definir al mismo tiempo en la pantalla un formato de SPRs de los 4 disponibles.
Entonces si uso SPRs de 16x16 normales, no puedo usar sprites de 8x8 o más grandes de 16x16?
Esto no es exactamente así. Si se selecciona este formato de SPRs puedes crear diferentes tamaños dentro de un SPR de 16x16 aunque sea de 8x8 pero después desde código tendrás que tener en cuenta el tamaño que has creado para el SPR, incluso se pueden usar 2 SPRs de 16x16 uno encima del otro para crear un SPR de 32x16 o puedes crearlos más anchos o mas grandes juntado varios SPR.



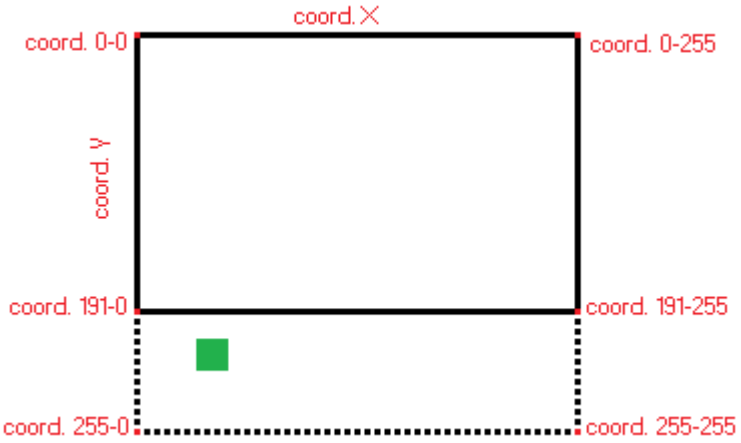
EJEMPLOS DE SPRs:

Como podéis ver no solo tenemos que limitarnos al tamaño de 16x16 se pueden hacer SPRs más grandes usando varios SPRs, y si desplazamos la posición de los SPRs podemos conseguir que den la sensación de ser mas grandes, o los puedes poner en la misma línea desplazándolos X pixeles como puede ver en las imágenes, eso ya depende de nuestra imaginación y de lo que queráis realizar. También podrías crearlos más pequeños de 8x8 dentro de un SPR de 16x16.

Vamos a explicar con más detalles cada uno de los valores de la tabla de ATRs de SPRs.

1º Valor – Coordenada Y - Esta es la coordenada Y del SPR su valor va desde 0 hasta 255.

2º Valor – Coordenada X - Esta es la coordenada X del SPR su valor va desde 0 hasta 255



Aquí te pongo una imagen para explicártelo La resolución del MSX en modo SC2 es de 256x192 pero el ordenador siempre cuenta desde 0, entonces es 255x191

Pero para los SPRs tiene una resolución de 255x255, como puedes ver en la imagen tenemos un SPR en la posición mas allá de la 191 que no se visualizaría en la pantalla, pero si podría estar en esa zona de la pantalla si su coordenada Y pasa de 192, en cuando superara la 255 empezaría a aparecer de nuevo al principio de la pantalla

Otra cosa que tienes que saber es que si sitúas un SPR en la Coordenada Y 208 dejaran de verse todos los SPRs que tengamos por encima de este plano. Y si lo sitúas en la Coordenada Y 209 dejara solo de verse ese SPR aunque no tiene mucho sentido ya que esta fuera de la pantalla y no lo vemos. Más cosas curiosas si sitúas un ATR en la coord. Y 8 realmente tienes que ponerlo en la coord.Y 7, al igual que si lo sitúas en la coord.Y 0 este se pondrá en la coord.Y 1, cosas extrañas del VDP.

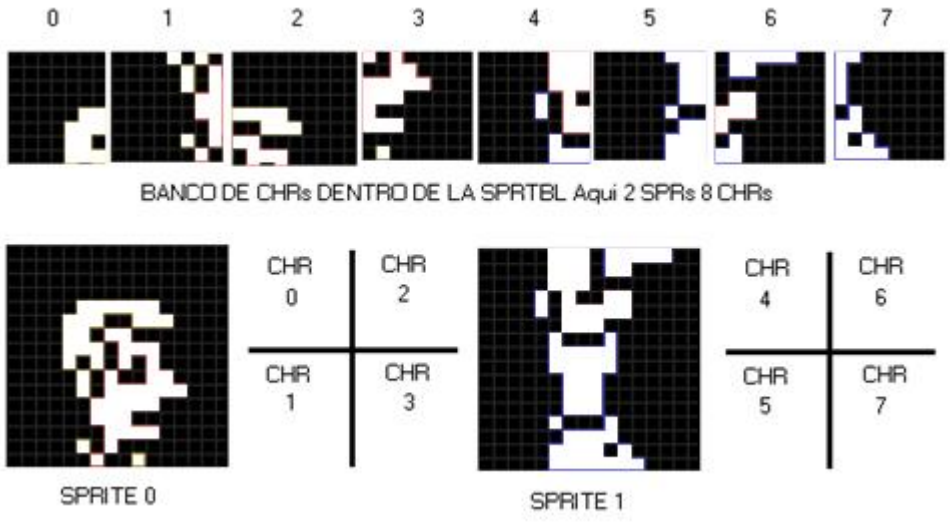
Planos de los sprites

Te explico el tema de los planos para que lo entiendas cada ATR que sitúas en la SPRATR va tomando un numero consecutivo de Plano empezando por el 0 y terminando por el 31. Ahora vamos a pensar en profundidad de planos el ATR1 sería el más cercano a la pantalla y el ATR32 sería el más alejado hacia el fondo de esta, te pongo una imagen que seguro que lo entiendes mejor, esto quiere decir que el ATR1 siempre seria el que pasa por encima de los demás sprites y así consecutivamente hasta llegar al ATR32 que sería el más alejado.

Sigamos con el tercer valor de la tabla de ATRs de SPRs.

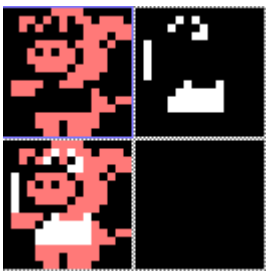
3º Valor – Numero de CHR – que CHR dentro de la SPRTBL usaremos para el SPR de 0 a 255

Al igual que ocurre con el banco de CHRs en la CHRTBL en la SPRTBL funciona de igual manera tenemos un banco de 256 CHRs numerados del 0 al 255 que son los gráficos de nuestros SPRs. Si hemos definido el VDP para usar SPRs de 8 x 8 cada CHR será un SPR pero si hemos definido que son de 16x16 cada SPR está compuesto de 4 CHR, esto quiere decir que aquí en este valor pondríamos 0 si queremos usar el SPR0 pero si queremos usar el SPR1 aquí tendríamos de decirle 4 que es donde empiezan los CHRs del SPR1. Vamos como siempre con las imágenes que lo aclaran más.



Sigamos con el cuarto valor de la tabla de ATRs de SPRs

4º Valor – Numero de Color – Que color de 0 a 15 de la paleta del MSX usara el SPR.



A cada SPR solo le podemos dar un color, en verdad son 2 colores el 0 que es el color transparente este no lo puedes modificar y el propio color que le das SPR de 0 a 15 de nuestra paleta MSX.
Entonces no puedo usar más de un color para un SPR? La respuesta es NO

No podemos usar más de un color por SPR pero si podemos poner un SPR encima de otro SPR para conseguir que nuestro SPR tenga 2 o más colores. Como podéis ver en la imagen 2 SPRs con 1 color y la mezcla de los 2 SPRs. Aunque debes saber que esto gastara 2 SPRs en la misma línea hay trucos para usar un tercer color poniendo un SPR más arriba de este SPR para que en la misma línea no coincidan y se produzca la terrible regla del 5º Sprite. (Y os preguntareis que es esto tan terrible)

Hay una limitación inherente al hardware que no podemos eliminar aunque si paliar un poco.

Me refiero al hecho de situar 5 SPRS en la misma línea horizontal, os recuerdo que siempre hablo de MSX1, en MSX2 esto mismo sucede pero con 8 SPRs en la misma línea. El problema es que si situamos 5 SPRs en la misma línea, el 5º SPR desaparecerá, bueno más bien que no se visualizara en la pantalla aunque si este en ese sitio. Tengo preparado un ejercicio para demostraros esto viéndolo en la pantalla de nuestro MSX más abajo.

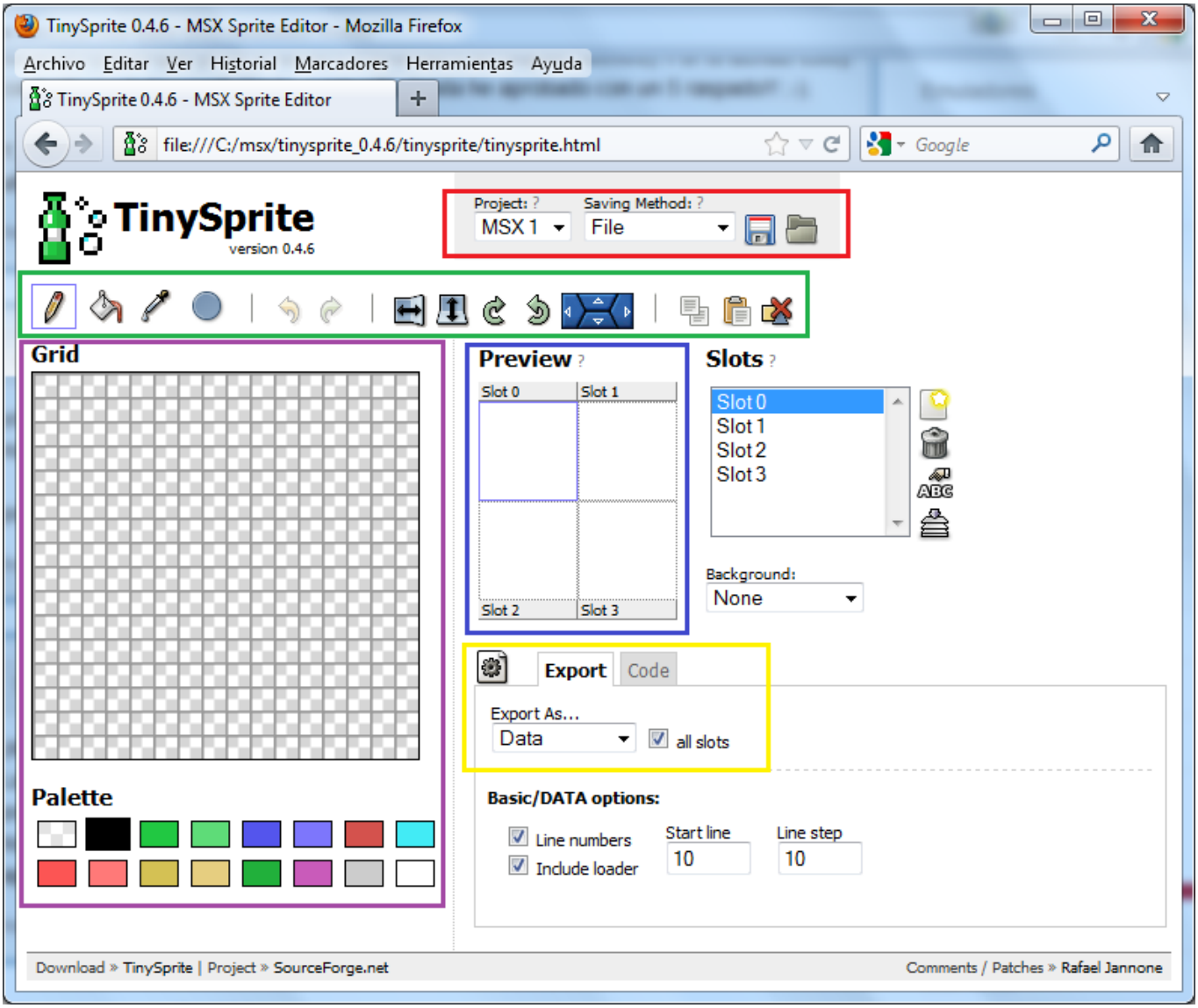
Y que se puede hacer para paliar esto… Pues hay 2 posibles soluciones.

1 – No utilizar más de cuatro SPRs en la misma línea horizontal.
2 – Usar trucos con los SPRs y crear una rutina de Parpadeo para paliar un poco este efecto.

Si se colocan estratégicamente los enemigos a diferentes alturas teniendo en cuenta 16 pixeles de distancia, podremos fácilmente saltar esta limitación porque nunca sucederá. Hay ocasiones en los que durante unos instantes se puede dar el caso de coincidir 5 SPRs en línea, entonces tendremos que crear por código una rutina de Parpadeo-Flickering. Lo que hace esta rutina es variar el orden de los SPRs en la SPRATR modificando los planos de los SPRs el 4º será el 1º el 1º será el 2º el 2º será el 3º y el 3º será el 4º repitiendo esta rotación constantemente lo que haremos es que durante unas decimas de segundo parpadee cada vez un SPR distinto, de esta manera el 5º SPR no desaparece pero vemos un pequeño parpadeo en los 5 SPRs, hasta que dejen de estar en la misma línea. Esto puede ser útil para un instante pero no para poner 5 SPRs en línea constantemente porque el efecto es molesto.

Aquí te pongo 2 capturas de ejemplos, En Zombie Incident la prota usa 3 SPRs, uno encima de otro y un 3º en medio de los 2 pero en línea solo tiene 4 los 2 de la prota mas los 2 del zombie verde. Yo en JumpinG uso 3 SPRs para el prota mas 2 SPR enemigos de 2 perros en la misma línea, eso son 5 pero yo opto porque no se visualice el 5º SPR que es la sombra negra del prota en vez de realizar por código parpadeo-Flickeo. Hay que jugar con estas cosas para salvar este Hándicap. Eso será elección tuya.

Vamos con la parte practica que la teoría ya la hemos pasado. Antes de hacer nuestro HolaSPR1.asm tenemos que crear el SPR que usaremos para nuestro ejercicio, la herramienta que yo utilizo para crear los SPRs está en el pack-MSX de la 1ª entrega y se llama el fichero comprimido tinysprite_0.4.6.zip descomprímelo en la carpeta C:\MSX donde tienes todas las herramientas, veras que te ha creado una carpeta llamada tinysprite_0.4.6 y dentro de esta carpeta otra carpeta llamada tinysprite, este programa creado por Rafael Jannone esta creado en JavaScript por eso veras que el ejecutable o donde tienes que pulsar doble clic es el fichero tinysprite.html pulsa y se abrirá con tu navegador web.(No I-explorer.)



Te he puesto unos rectángulos de colores para enseñarte un poco como funciona este programa que aunque es muy simple usarlo no está de más explicarlo.

1º - Rojo - Aquí seleccionamos SPRs para MSX1 o 2 además de grabar o cargar un SPR.

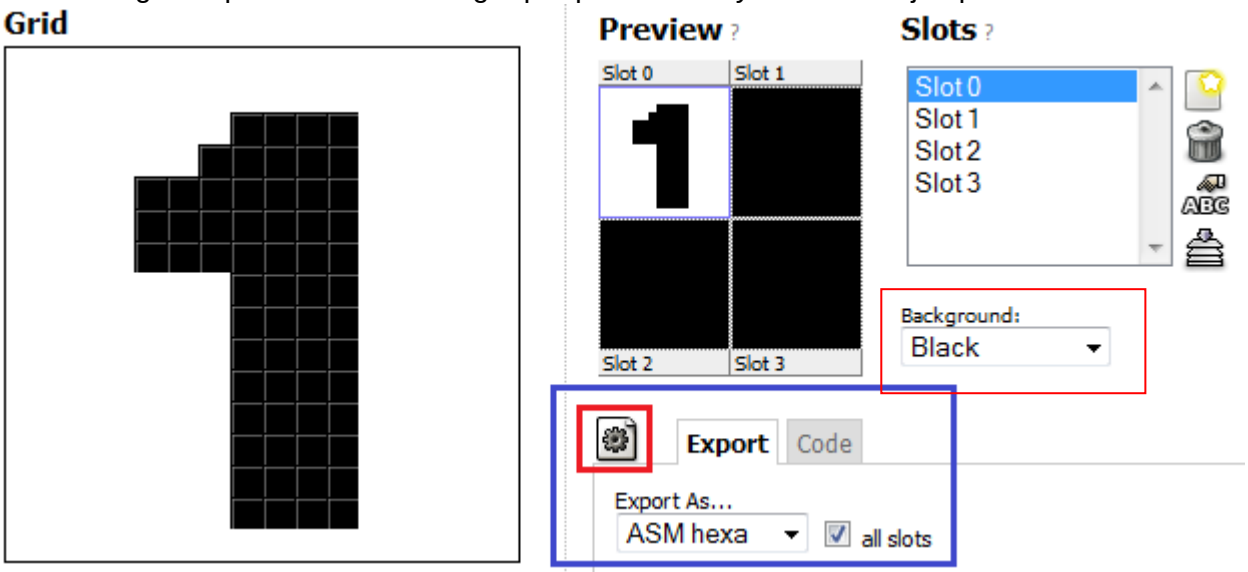
2º - Verde – como en muchos programas de dibujo lápiz para pintar al pixel, relleno, leer un color, crear círculos, Undo y restore, Flip Horizontal o vertical, rotación, desplazar la ventana, copiar pegar y borrar.

3º - Morado - Es la zona de la pantalla donde dibujamos el SPR y donde seleccionamos los colores a usar. Tengo que decirte que cada color que uses aquí seria un SPR distinto.

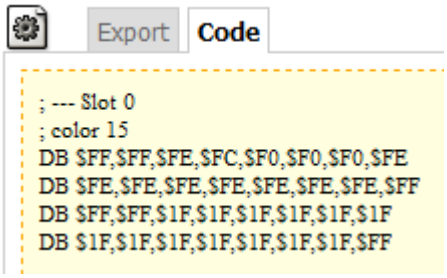
4º - Color Azul – Es la zona donde vemos en miniatura cada uno de los 4 posibles SPRs a crear.

5º - Color Amarillo – Sin lugar a dudas es la zona más importante, junto con la de leer y grabar porque aquí es donde nos genera los DB´s de los bytes de los SPRs a formato ASM u otros.

Vamos a crear nuestro primer SPR lo primero que selecciono es en el ComboBox “Background” el color “Black” me gusta que el fondo sea negro porque en la mayoría de los ejemplos uso este color de fondo.

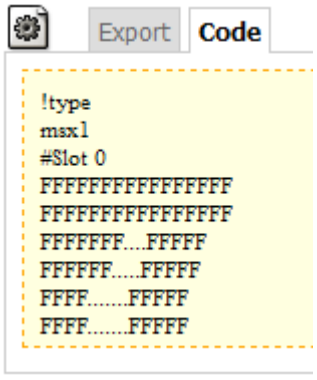


Después creo un SPR cuadrado haciendo un fill de todo el SPR en blanco y vacio con tinta transparente NO con negro el SPR número 1, esto es para que se vea bien el SPR de 16x16 cuadrado y en interior del numero deje ver lo que hay de detrás de la pantalla. Remarcado en Azul - En el ComboBox Export_As… aquí seleccionamos en que formato vamos a exportar los datos. Selecciona ASM hexa finalmente desmarca el CheckBox que en la imagen ves con una V opción all slots para que solo exporte el Slot 0 y no los 4 Slots, o posibles SPRs y pulsáis sobre el recuadro en rojo botón Export.

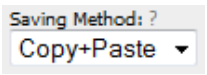


Esto es lo que nos genera el tiny que no es otra cosa que los bytes de los 4 CHRs que componen nuestro primer SPR, como es texto podemos copiarlo y pegarlo en nuestro código. Datos que te da el tiny, de que Slot es el SPR, en este caso solo el Slot 0, y que color usa el SPR en este caso el 15 que es blanco a modo de comentario. Si creas un SPR con 2 CLRs te duplicaría esta información como te he explicado en la teoría porque serian 2 SPRs y tu sabrías cual es un SPR y cuál es el otro porque dentro del Slot 0 te pondría 32 bytes para el 1º color y 32 bytes para el 2º color. Vamos a ver como guardar este SPR en el tiny.

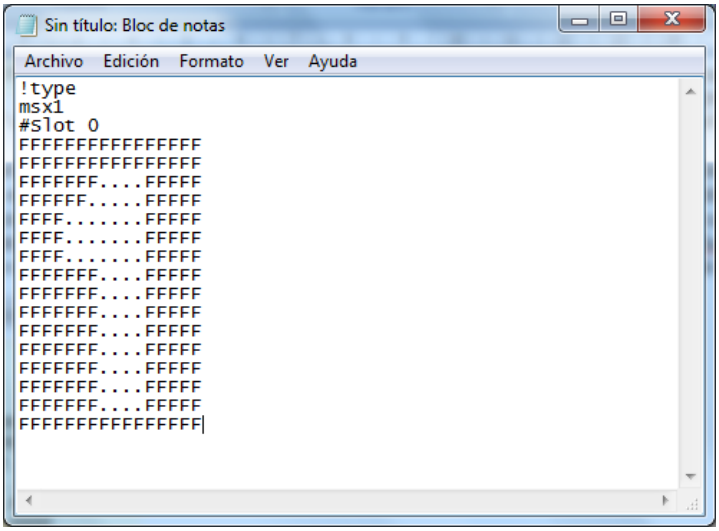
En el apartado 1º el Rojo del tinysprite tienes el botón Save os pongo la imagen a la derecha. Pulsando el botón y después diciéndole el nombre guardaría el fichero pero hay un problema. He de deciros que a mí no me funciona después la opción de Load y nunca puedo cargar el fichero por eso uso otra opción para grabar mis proyectos de SPRs, os lo explico.



En el ComboBox Saving Method:



cambio la opción de File por la opción de Copy+Paste y pulso el botón Save veras la imagen de la IZQ..



Ahora abro el Bloc de notas de Windows y copio y pego el texto del tiny al bloc de notas y guardo el fichero como SPR1_paste.txt
Para cargarlo de nuevo abro el tiny siempre con la opción Copy-Paste pulso el botón Load y en la ventana que se abre pego el texto del bloc de notas y pulso el botón Import y ya me sale mi SPR.

Vamos con el código del HolaSPR1.asm abrir el EditPlus y creáis un fichero nuevo ASM, copia y pega todo el texto a partir de esta línea en el fichero nuevo que habéis creado en el EditPlus.

```
;-------------------------------------------------------
; Nombre de nuestro programa
; Hola Sprite 1
; 17/10/2011
;-------------------------------------------------------

;-------------------------------------------------------
; CONSTANTES
;-------------------------------------------------------
       NUM_SPR        equ     1       ; Numero de SPRITES
       NUM_ATR        equ     1       ; Numero de Atributos de Sprite

;-------------------------------------------------------
; VARIABLES
;-------------------------------------------------------
; Direcciones de la VRAM
       CHRTBL         equ     0000h   ; Tabla de caracteres
       NAMTBL         equ     1800h   ; Tabla de Patrones
       CLRTBL         equ     2000h   ; Tabla del color de los caracteres
       SPRTBL         equ     3800h   ; Tabla de Sprites
       SPRATR         equ     1B00h   ; Tabla de los atributos de los sprites
; Variables de sistema
       CLIKSW         equ     $F3DB   ; Keyboard click sound
       FORCLR         equ     $F3E9   ; Foreground colour
       RG1SAV         equ     $F3E0   ; Content of VDP(1) register (R#1)
       STATFL         equ     $F3E7   ; Content of VDP(8) status register (S#0)

;-------------------------------------------------------
; DIRECTIVAS PARA EL ENSAMBLADOR ( asMSX )
;-------------------------------------------------------
       .bios          ; Definir Nombres de las llamadas a la BIOS
       .page 2        ; Definir la dirección del código irá en 8000h
       .rom           ; esto es para indicar que crearemos una ROM
       .start INICIO  ; Inicio del Código de nuestro Programa
; Seguir la norma del Standard MSX
       dw 0,0,0,0,0,0 ; 12 ceros
; Identificacion
       db "HolaSPR1",1Ah

;-------------------------------------------------------
; INICIO DEL PROGRAMA
;-------------------------------------------------------
INICIO:
       call INIT_MODE_SCx    ; iniciar el modo de pantalla
       call INIT_GRAFICOS    ; imprimir el mensaje en pantalla

; Mostrar nuestro primer sprite
; Mover los ATRs de los SPRs en ROM/RAM a la SPRTBL en VRAM
       ld     hl,tblATRsSPRs ; Tabla de los ATRs de los SPRs en ROM
       ld     de,SPRATR      ; Destino la Sprite Atribute Table
       ld     bc,(NUM_ATR*4) ; Numero de Bytes
       call   LDIRVM         ; 05Ch - BIOS - Copy block to VRAM, from memory

FIN:
       jp     FIN            ; esto es como 100 goto 100

;-------------------------------------------------------
INIT_MODE_SCx:
; INICIALIZA EL MODO DE PANTALLA Y COLORES
;-------------------------------------------------------
; BASIC: COLOR 15,1,1
; Establecer los colores
       ld     hl,FORCLR      ; Variable del Sistema
       ld     [hl],15        ; Color del primer plano 15=blanco
       inc    hl             ; FORCLR+1
       ld     [hl],1         ; Color de fondo 1=negro
       inc    hl             ; FORCLR+2
       ld     [hl],1         ; Color del borde 1=negro
; call INITXT          ; 06Ch - BIOS - Initialize VDP to 40x24 Text Mode - set SCREEN 0
; call INIT32          ; 06Fh - BIOS - Initialize VDP to 32x24 Text Mode - set SCREEN 1
       call   INIGRP         ; 072h - BIOS - Initialize VDP to Graphics Mode - set SCREEN 2
; call INIMLT          ; 075h - BIOS - Initialize VDP to Multicolour Mode - set SCRREN 3
;
; SCREEN 0 : texto de 40 x 24 con 2 colores
; SCREEN 1 : texto de 32 x 24 con 16 colores
; SCREEN 2 : graficos de 256 x 192 pixeles con 16 colores
; SCREEN 3 : graficos de 64 x 48 pixeles con 16 colores
;
```

```
; Esto es para quitar el sonido que emite el msx cuando se pulsa una tecla
        xor     a               ; ld a,0
        ld      [CLIKSW],a      ; Variable BIOS desactivar sonido teclas

; Desactivar las teclas de funcion
        call    ERAFNK          ; 0CCh - BIOS - Erase function key display

; Modificar el Registro 1 del VDP ( Video Display Procesor )
;
; bit 7 : 10000000b: 1= siempre a 1 (4/16k vram)
; bit 6 : 01000000b: 1= Pantalla activada | 0= Pantalla desactivada
; bit 5 : 00100000b: 1= interrupciones activadas | 0= interrupciones desactivadas
; bit 4 : 00010000b: 1= modo texto | 0= otros modos
; bit 3 : 00001000b: 1= modo multicolor | 0= otros modos
; bit 2 : 00000000b: 0= siempre a 0
; bit 1 : 00000010b: 1= sprites de 16x16 | 0= sprites de 8x8
; bit 0 : 00000001b: 1= sprites *2 Ampliados | 0= sprites *1 no ampliados

;       ld      bc,(11100010b<<8)|1    ; BC=E201h en binario

        ld      bc,$E201        ; B=E2 en binario es 11100010b  y  C=01  que es el registro del VDP
        call    WRTVDP          ; 047h - BIOS - Write to any VDP register
        ret                     ; salir de la rutina
;-----------------------------------------------------------

;-----------------------------------------------------------
INIT_GRAFICOS:
; COLOCA LOS GRAFICOS EN LA VRAM
;-----------------------------------------------------------
; Esto lo realizamos para que no se vea nada en la pantalla
        call    DISSCR          ; 041h - BIOS - Disable screen

; Borrar los 768 Bytes de la tabla de nombres en VRAM
        ld      hl,NAMTBL       ; Origen la Name Table
        ld      bc,768          ; Numero de bytes 32x24
        xor     a               ; ld a,0 - Valor a rellenar
        call    FILVRM          ; 056h - BIOS - Fill block of VRAM with data byte

; Colocar los bytes de los SPRs en la Sprite Pattern Table
        ld      hl,SPR1         ; Origen = Bytes de nuestro sprite
        ld      de,SPRTBL       ; Destino = Sprite Pattern Table
        ld      bc,(NUMSPR*32)  ; Numero de bytes
        call    LDIRVM          ; 05Ch - BIOS - Copy block to VRAM, from memory

; Esto lo realizamos para que se muestre de nuevo la pantalla
        call    ENASCR          ; 044h - BIOS - Enable screen

        ret                     ; salir de la rutina INIT_GRAFICOS
;-----------------------------------------------------------

;-----------------------------------------------------------
; Tabla de los atributos de los sprites
tblATRsSPRs:
        db      86,120,  0,  8   ; Coord.Y,Coord.X,n°SPR,Color
;-----------------------------------------------------------

;-----------------------------------------------------------
; Sprite Numero 1
; generated by TinySprite
SPR1:
; --- Slot 0
; color 15
DB $FF,$FF,$FE,$FC,$F0,$F0,$F0,$FE
DB $FE,$FE,$FE,$FE,$FE,$FE,$FE,$FF
DB $FF,$FF,$1F,$1F,$1F,$1F,$1F,$1F
DB $1F,$1F,$1F,$1F,$1F,$1F,$1F,$FF
```

Aquí finaliza el código de nuestro HolaSPR1.asm copia todo menos esta línea.

Ahora como siempre vamos a explicar el código, estoy seguro que a medida que avanzáis con los tutoriales ya sabéis lo que hace la mayor parte de este código. Así que solo describiré lo nuevo.

La cabecera de nuestro código como siempre nombre y fecha de nuestro proyecto. Direcciones etc.

Esta es la primera vez desde que escribo los tutoriales que uso la sección de CONSTANTES, el que sabe de programación seguro que sabe lo que es esto, al igual que una variable. Esto le dice a nuestro código que el nombre NUM_SPR = 1 y NUM_ATR = 1 después veremos porque lo usamos.

```
;-------------------------------------------------------
; CONSTANTES
;-------------------------------------------------------
        NUM_SPR         equ     1       ; Numero de SPRITES
        NUM_ATR         equ     1       ; Numero de Atributos de Sprite
```

Ahora en la sección donde defino las variables del MSX he agregado 2 nuevas variables si las miras en el fichero de VariablesMSX.txt del pack-MSX te dará más información, pero te lo resumo, cada registro del VDP es almacenado en las variables del sistema MSX desde $F3E0 hasta $F3E6 y el registro de estado del VDP en $F3E7 después más adelante veremos porque las he incluido.

```
        RG1SAV          equ     $F3E0   ; Content of VDP(1) register (R#1)
        STATFL          equ     $F3E7   ; Content of VDP(8) status register (S#0)
```

Esto es una cosa que no es necesaria, pero es cómoda para cuando miras un fichero ROM por dentro para saber que versión es, ya que este texto se mostrara como ASCII al ver la ROM con un editor.

```
; Identificacion
        db "HolaSPR1",1Ah
```

En nuestra querida rutina INIT_MODE_SCx he agregado 2 cosas nuevas, ya sabes que esta es la rutina encargada de seleccionar el modo de video SCREEN 2, ya en ultimo tutorial vimos que añadí que se desactive el sonido de las teclas y ahora vamos a ver 2 cosas nuevas que he agregado a esta rutina.

La 1ª creo que poco hay que comentar pero aquí va, el BASIC activa unas teclas de función esas que sacan al pulsar las teclas F1 a F12 las palabras clave en BASIC como COLOR, RUN , etc. etc. Pues esta llamada a la BIOS lo que hace es desactivar todas estas teclas de función.

```
; Desactivar las teclas de funcion
        call    ERAFNK          ; 0CCh - BIOS - Erase function key display
```

La 2ª y la más importante para la zona del tutorial donde estamos es la selección del modo de SPRs que vamos a utilizar y que te explique en la teoría, SPR de 8x8 o 16x16 normales o ampliados.

```
        ld      bc,$E201        ; B=E2 en binario es 11100010b  y  C=01  que es el registro del VDP
        call    WRTVDP          ; 047h - BIOS - Write to any VDP register
```

Esto es lo único que tendrías que hacer para seleccionar SPRs de 16x16 no ampliados. Pero esto no puede quedar sin explicación y por eso en el código he añadido otras formas de hacerlo, incluso la explicación que ahora mismo voy a detallarte del porque de las cosas.
No sé si lo sabéis o no, pero el VDP es el procesador grafico encargado de manejar la VRAM, toda la información grafica y los sprites del MSX, este VDP el TMS9918 tiene 7 tiene Registros y uno de ESTADO para realizar el acceso a este procesador. No me voy a enrollar que esto no es el libro rojo del MSX. Pero si hay cosas básicas que hay que saber, y una de ellas es el modo de los SPRs.

En el registro 1 del VDP se manejan estos bits.

```
; Modificar el Registro 1 del VDP ( Video Display Procesor )
;
; bit 7 : 10000000b: 1= siempre a 1 (4/16k VRAM)
; bit 6 : 01000000b: 1= Pantalla activada | 0= Pantalla desactivada
; bit 5 : 00100000b: 1= interrupciones activadas | 0= interrupciones desactivadas
; bit 4 : 00010000b: 1= modo texto | 0= otros modos
; bit 3 : 00001000b: 1= modo multicolor | 0= otros modos
; bit 2 : 00000000b: 0= siempre a 0
; bit 1 : 00000010b: 1= sprites de 16x16 | 0= sprites de 8x8
; bit 0 : 00000001b: 1= sprites *2 Ampliados | 0= sprites *1 no ampliados
```

Cuando llamamos al la BIOS a la rutina DISSCR realmente lo que hace esta rutina es modificar el BIT 6 del registro del VDP para desactivar la pantalla. Pero a nosotros en este registro lo que nos interesa son los BITs 0 y 1 si te fijas en la información técnica extraída del RED BOOK del MSX que he puesto a modo de comentarios, si el BIT 1 del Registro 1 del VDP los ponemos a 1 selecciona SPRs de 16x16 y si lo ponemos a 0 selecciona SPRs de 8x8, si modificamos el BIT 0 a 1 los SPRs serán ampliados y si lo ponemos a 0 serán normales. Puedes hacer 2 cosas o leer el registro 1 del VDP y modificar solo los BITs que nos interesa, o bien meter a saco el valor, siempre teniendo en cuenta los valores de los 8 bits del registro 1, según la tabla de bits que te he puesto a modo de comentarios.

Veamos de donde sale el valor $E201 que ponemos en ld bc,$E201 como BC es un registro de 16 Bits lo separamos y tenemos el Registro C=01 y el Registro B=E2 y llamamos a la rutina en BIOS WRTVDP a esta rutina se la llama con el número del registro del VDP que vamos a modificar en C y el valor a

escribir en el registro en B y llamamos con call WRTVDP y esta escribirá el valor E2 en el registro 1 del VDP.(Seguro que te preguntas que es el valor E2) abre la calculadora de Windows y en el menú ver selecciona el formato Programador. Ahora convierte el valor E2 de hexadecimal a binario y viendo los 0 y los 1 sabrás con la información de arriba de los BITs que hace este valor.
E2 en binario es 11100010 lo ponemos en vez de izquierda a derecha de arriba a abajo para explicarlo.

1-Bit 7 – Siempre a 1
1-Bit 6 – Pantalla activada
1-Bit 5 – Interrupciones activadas
0-Bit 4 – Como es Screen 2 OTRO MODO
0-Bit 3 – Como es Screen 2 OTRO MODO
0-Bit 2 – Siempre a 0
1-Bit 1 – Sprites de 16x16
0-Bit 0 – No ampliados

Por este valor nuestros sprites se mostraran en pantalla con el tamaño de 16x16 no ampliados.

Os recomiendo esta forma de cambiar el registro 1 del VDP igual que ponemos ld bc,$E201 se puede poner también de esta forma que os pongo debajo de este texto, donde puedes ver claramente los 8 bits y el ultimo valor el 1 que sería el registro a modificar en nuestro caso el registro 1. Para el Ejercicio1 del final del tutorial deberás activar esta opción en el código quitando el ; en esta opción que te explico y poniendo el ; antes del ld bc,$E201 para anular esta línea y dejarla como un comentario.

```
        ld      bc,(11100010b<<8)|1     ; BC=$E201 en binario
        call    WRTVDP                  ; 047h - BIOS - Write to any VDP register
```

Con esto ya hemos terminado de explicar los cambios en la rutina INIT_MODE_SCx

Vamos con la rutina INIT_GRAFICOS otra vieja conocida que seguro que ya sabéis lo que hace, desactivamos la pantalla, borramos la NAMTBL, colocamos los gráficos en VRAM y mostramos de nuevo la pantalla, aquí usamos como siempre la llamada a la rutina en BIOS LDIRVM.para mover bytes de ROM/RAM a VRAM. Moviendo los bytes del SPR1 creado con el tiny a la Sprite Pattern Tabble.

```
; Colocar los bytes de los SPRs en la Sprite Pattern Table
        ld      hl,SPR1                 ; Origen = Bytes de nuestro sprite
        ld      de,SPRTBL               ; Destino = Sprite Pattern Table
        ld      bc,(NUM_SPR*32)         ; Numero de bytes
        call    LDIRVM                  ; 05Ch - BIOS - Copy block to VRAM, from memory
```

Ya sabeis que que cada SPR son 4 CHRs y como un CHR son 8 Bytes pues 4*8=32 que son los bytes que ocupa un SPR como hemos definido una CONSTANTE esta NUM_SPR=1 pues multiplicamos 1*32 ahora que tenemos un solo SPR pero a medida que vayamos incorporando mas SPRs al código no tendremos que cambiar esta parte del código, solo poner los bytes de otros SPRs y modificar la variable NUM_SPR cambiando su valor por el numero de SPRs que vayamos agregando al código.
Vamos con otra parte explicada en la teoría de este tutorial referente al apartado donde explico los Atributos de los Sprites (ATRs de los SPRs)

Esta es la tabla que hemos definimos en el código y que mostrara nuestro primer SPR.

```
;-----------------------------------------------------------
; Tabla de los atributos de los sprites
ATRsSPRs:
        db      86,120,  0,  8   ; Coord.Y,Coord.X,n°SPR,Color
;-----------------------------------------------------------
```

Vamos a colocar un SPR en las Coordenadas Y=86 y X=120 de la pantalla que es el centro. Usaremos el SPR1 que empieza en el CHR 0 y le daremos el color Rojo=8

Después de esta tabla es donde tenéis que pegar los bytes en formato DB´s que hemos dibujado con el programa TinySprite y que hemos exportado a formato ASM añadiéndole la etiqueta SPR1:

```
;-----------------------------------------------------------
; Sprite Numero 1
; generated by TinySprite
SPR1:
; --- Slot 0
; color 15
DB $FF,$FF,$FE,$FC,$F0,$F0,$F0,$FE
DB $FE,$FE,$FE,$FE,$FE,$FE,$FE,$FF
DB $FF,$FF,$1F,$1F,$1F,$1F,$1F,$1F
DB $1F,$1F,$1F,$1F,$1F,$1F,$1F,$FF
```

Llegados a este punto en el que tenemos los CHRs de nuestro SPR en la SPRTBL, solo nos falta que ese SPR se muestre en la pantalla, por eso he agregado en el bucle principal del Ejemplo este código.

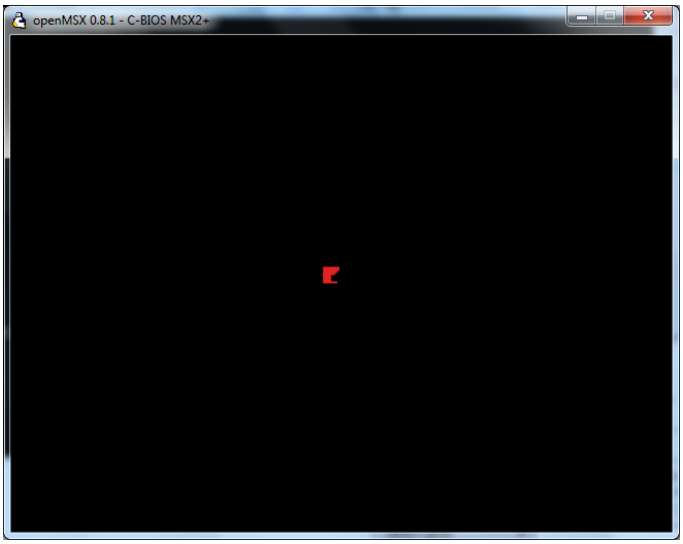
```
; Mostrar nuestro primer sprite
; Mover los ATRs de los SPRs en ROM/RAM a la SPRTBL en VRAM
        ld      hl,tblATRsSPRs ; Tabla de los ATRs de los SPRs en ROM
        ld      de,SPRATR      ; Destino la Sprite Atribute Table
        ld      bc,(NUM_ATR*4) ; Numero de Bytes
        call    LDIRVM         ; 05Ch - BIOS - Copy block to VRAM, from memory
```

Esto leerá los bytes de la tabla tblATRsSPRs y los colocara en la VRAM en la SPRATR. Ya sabes que que cada ATR de SPR son 4 valores como hemos definido una CONSTANTE esta NUM_ATR=1 pues multiplicamos 1*4 ahora que tenemos un solo ATR pero a medida que vayamos incorporando mas ATRs al código no tendremos que cambiar esta parte del código, solo poner las tablas de otros ATRs y modificar la variable NUM_ATR cambiando su valor por el numero de ATRs que vayamos agregando al código. Este será el SPR1 en el Plano 0, ya que es nuestro primer SPR en esta zona de la VRAM.

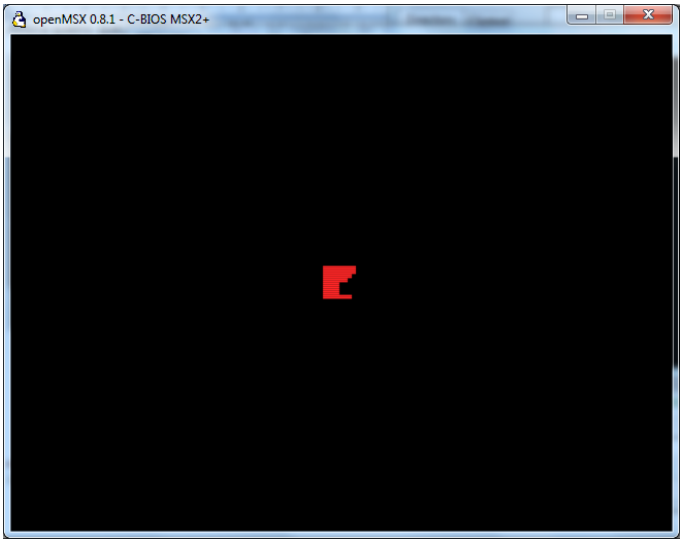
Ya estáis preparado para compilar y ejecutar la ROM de nuestro nuevo ejemplo el resultado tiene que ser la primera imagen de este tutorial.

EJERCICIO 1:

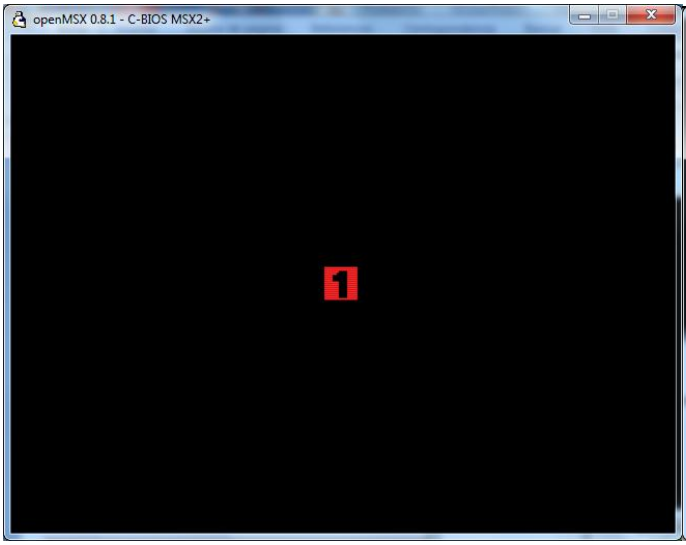
Ahora quiero que modifiquéis los BITs del registro 1 del VDP para que veas cómo seleccionar los diferentes modos de los SPRs compila y ejecuta cada cambio en la ROM de este ejemplo para que veas el resultado que se produce te adjunto las 4 capturas con los 4 formatos.



Sprites de 8x8 pixeles no ampliados



Sprites de 8x8 pixeles Ampliados



Sprites de 16x16 pixeles no ampliados

Sprites de 16x16 pixeles Ampliados

EJERCICIO 2:

Con lo que habéis aprendido podéis poner 2 SPRs en la pantalla usando el mismo grafico.

En la sección de CONSTANTES modificáis el valor 1 por un 2 en NUM_ATR ya que vamos a añadir un ATR nuevo sin añadir otro grafico, de esta manera no habrá que modificar código.

```
NUM_ATR         equ     2       ; Numero de Atributos de Sprites
```

Ahora añadir una segunda tabla de ATRs para nuestro SPR2

```
;-----------------------------------------------
; Tabla de los atributos de los sprites
tblATRsSPRs:
        db      86,120,  0,  8    ; Coord.Y,Coord.X,n°SPR,Color
        db      86,144,  0,  3    ; Coord.Y,Coord.X,n°SPR,Color
;-----------------------------------------------
```

Podéis ver en la segunda tabla, que añado la misma Coordenada Y, pero en la Coordenada X le sumo 24 para que se vea a la derecha del primero dejando 8 pixeles de separación entre el SPR1 y el SPR2 el numero de SPR lo dejo igual en 0 ya que solo tenemos 1 grafico, aunque ahora le pongo color Verde 3 al SPR. Compila y lanza la ROM el resultado será la imagen pequeña que ves encima de este texto.

Ahora vamos a ver sobre la pantalla del MSX lo que te explique en la teoría sobre los planos y la regla del 5º Sprite viéndolo con código y sobre una maquina real o emulador.

EJERCICIO 3:

Vamos con el orden de los planos, en el EditPlus vamos a File – Save as… Y el HolaSPR1.asm lo vamos a grabar como HolaSPR2.asm para añadirle las siguientes modificaciones.

En la sección CONSTANTES tiene que quedar así.

```
NUM_SPR         equ     5       ; Numero de SPRITES
NUM_ATR         equ     5       ; Numero de Atributos de Sprite
```

Ahora en la tabla de ATRs de SPRs tiene que quedar como ves aquí.

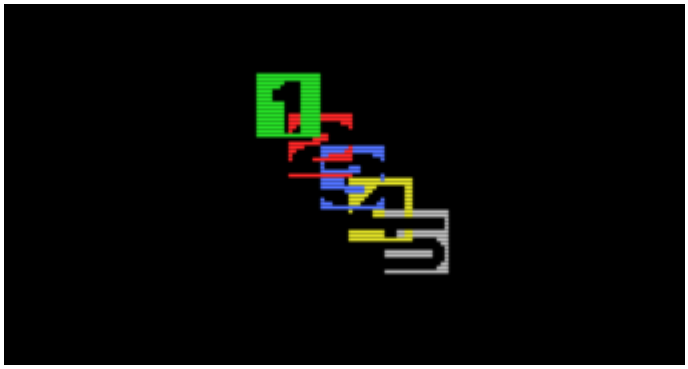
```
;-----------------------------------------------
; Tabla de los atributos de los sprites
tblATRsSPRs:
        db      86,120,  0,  2    ; Coord.Y,Coord.X,n°SPR,Color
        db      96,128,  4,  8    ; Coord.Y,Coord.X,n°SPR,Color
        db     104,136,  8,  5    ; Coord.Y,Coord.X,n°SPR,Color
        db     112,143, 12, 10    ; Coord.Y,Coord.X,n°SPR,Color
        db     120,152, 16, 14    ; Coord.Y,Coord.X,n°SPR,Color
;-----------------------------------------------
```

Ahora he creado mas SPRs con el tinysprite poniendo números del 1 al 5, Esto son sus bytes.
Que teneis que sustituir por el del número 1 que tenemos ahora mismo en el código.

```
;----------------------------------
; Sprite Numeros 1 al 5
; generated by TinySprite
SPR1:
; --- Slot 0
; color 15
DB $FF,$FF,$FE,$FC,$F0,$F0,$F0,$FE
DB $FE,$FE,$FE,$FE,$FE,$FE,$FE,$FF
DB $FF,$FF,$1F,$1F,$1F,$1F,$1F,$1F
DB $1F,$1F,$1F,$1F,$1F,$1F,$1F,$FF
SPR2:
; --- Slot 0
; color 15
DB $FF,$FF,$F0,$C0,$80,$81,$03,$FF
DB $F0,$C0,$80,$83,$00,$00,$00,$FF
DB $FF,$FF,$07,$01,$00,$C0,$C0,$00
DB $01,$03,$3F,$FF,$00,$00,$00,$FF
SPR3:
; --- Slot 1
; color 15
DB $FF,$FF,$F0,$C0,$80,$81,$FF,$FC
DB $FC,$FC,$FF,$03,$00,$80,$E0,$FF
DB $FF,$FF,$07,$01,$00,$C0,$C0,$01
DB $01,$00,$E0,$E0,$00,$01,$03,$FF
```

```
SPR4:
; --- Slot 2
; color 15
DB $FF,$FF,$FE,$FC,$F8,$F0,$E0,$C1
DB $83,$03,$00,$00,$00,$FF,$FF,$FF
DB $FF,$FF,$03,$03,$03,$03,$83,$83
DB $83,$83,$00,$00,$00,$83,$83,$FF
SPR5:
; --- Slot 3
; color 15
DB $FF,$FF,$00,$00,$00,$1F,$1F,$00
DB $00,$00,$FF,$FF,$00,$00,$00,$FF
DB $FF,$FF,$01,$01,$01,$FF,$FF,$07
DB $03,$01,$F1,$F1,$01,$03,$07,$FF
```

Ahora que habéis incluido todo compila y lanza la ROM.



Aquí podéis ver que el SPR1 está en el plano 0 al ser el primer ATR y así sucesivamente.

Podéis ver con más claridad como cada SPR esta encima del otro SPR según el orden de los planos, que a su vez es el mismo orden en el que hemos ido situando cada ATR en la SPRATR.

EJERCICIO 4:

Vamos a ver otra característica del VDP al situar un SPR en la Coordenada Y 208, recuerda que te explique en la teoría de este tutorial que si pones un SPR en esa coordenada todos los SPRs del plano inferior a este dejaran de verse, vamos a comprobarlo. Esta es una forma de ocultar los SPRs en la pantalla en cualquier momento que nos interese realizar esta acción.

```
;-----------------------------------------------
; Tabla de los atributos de los sprites
tblATRsSPRs:
        db     86,120,  0,  2   ; Coord.Y,Coord.X,n°SPR,Color
        db     96,128,  4,  8   ; Coord.Y,Coord.X,n°SPR,Color
        db    104,136,  8,  5   ; Coord.Y,Coord.X,n°SPR,Color
        db    112,143, 12, 10   ; Coord.Y,Coord.X,n°SPR,Color
        db    120,152, 16, 14   ; Coord.Y,Coord.X,n°SPR,Color
;-----------------------------------------------
```

Modificáis la coordenada Y del 1º ATR y donde pone 86 cambiarlo por un 208. Compila y lanza la ROM. El resultado es que no se verá ningún SPR en la pantalla. Vuelve a poner el 86 al ATR 1 y cambia el 96 del 2º ATR por un 208. Compila y lanza la ROM. Aquí puede ver que el SPR1 si se ve pero el resto no porque todos los planos inferiores al 1 no se ven cuando sitúas un SPR en la coordenada. Y 208.
Si queréis probar con el 3º ATR ya es cosa vuestra.

EJERCICIO 5:

Vamos a ver qué pasa cuando ponemos 5 SPRs en la misma línea horizontal.
Modificáis la tabla de ATR de esta manera y compila y lanza la ROM.

```
;-----------------------------------------------
; Tabla de los atributos de los sprites
tblATRsSPRs:
        db     24, 24,  0,  2   ; Coord.Y,Coord.X,n°SPR,Color
        db     24, 48,  4,  8   ; Coord.Y,Coord.X,n°SPR,Color
        db     24, 72,  8,  5   ; Coord.Y,Coord.X,n°SPR,Color
        db     24, 96, 12, 10   ; Coord.Y,Coord.X,n°SPR,Color
        db     24,120, 16, 14   ; Coord.Y,Coord.X,n°SPR,Color
;-----------------------------------------------
```

Como puedes ver en la imagen el 5º SPR está al lado del 4 pero no se ve.

EJERCICIO 6:

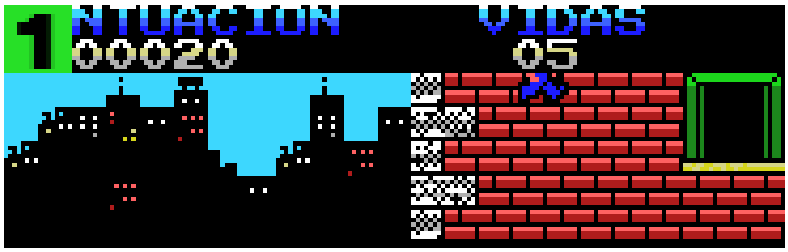
Vamos a mover 2 pixeles más abajo cada SPR para que no coincidan todos en la misma línea.

```
;-------------------------------------------------
; Tabla de los atributos de los sprites
tblATRsSPRs:
        db      24, 24,  0,  2    ; Coord.Y,Coord.X,n°SPR,Color
        db      26, 48,  4,  8    ; Coord.Y,Coord.X,n°SPR,Color
        db      28, 72,  8,  5    ; Coord.Y,Coord.X,n°SPR,Color
        db      30, 96, 12, 10    ; Coord.Y,Coord.X,n°SPR,Color
        db      32,120, 16, 14    ; Coord.Y,Coord.X,n°SPR,Color
;-------------------------------------------------
```

Una vez que modifiquéis la tabla de ATRs compila y lanza la ROM.
Como podéis ver ahora si se muestra parcialmente el SPR nº5. (Porque se muestra parcialmente?)
La respuesta está en que la norma del 5º SPR solo se cumple cuando hay 5 SPR en la misma línea horizontal. Fíjate en la imagen que te pongo a continuación.

Pongo 2 líneas imaginarias para que veas que la parte que desaparece del SPR5 es justo la que coincide con el SPR1 esto sucede porque esa parte del SPR5 es la única zona donde coinciden los 5 SPRs en la misma línea.

TRUCO en JumpinG. Una forma de sacar provecho a nuestro favor de la regla del 5º SPRs.

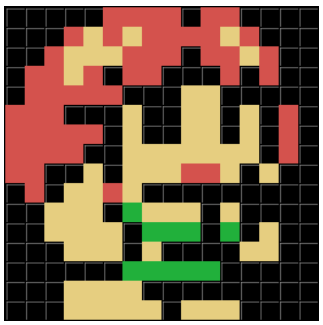


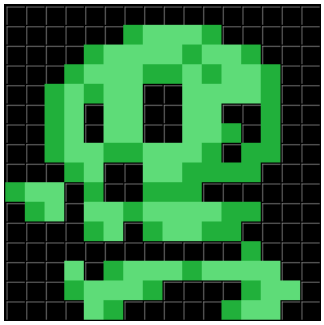
He puesto el SPR1 en la imagen para explicarte este truco
Te pongo una fragmento de mi juego JumpinG, si os fijáis cuando el prota salta y pasa por encima de los marcadores aquí puedes ver los pies, la cabeza no pasa por encima del número de vidas.
Realmente si lo está haciendo pero aquí se produce la regla del 5º SPR desapareciendo la parte del prota cuando coinciden 5 SPR en la misma línea, esto se consigue colocando 4 SPR con color transparente donde está el numero 1, estos SPRs no se ven porque su color es transparente y están en los planos 0,1,2,3 después sitúo el prota en el plano 5, así cuando el prota salta por encima del marcador al haber 5 SPRs en la misma línea, desaparece esta parte del prota dejando ver el marcador.

TRUCO del Zombie Incident para salvar la regla del 5º Sprite.

Fijaros en estas imágenes del juego Zombie Incident

La Prota y el Zombie Verde de la imagen
Os pongo un Zoom de los 2 para la explicación.

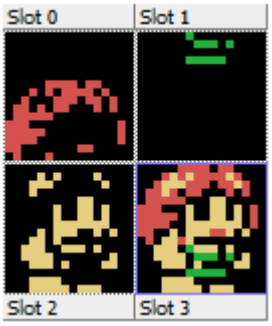




La prota tiene 3 colores que son 3 SPRs mas 2 colores del Zombie verde otros 2 SPRs esto quiere decir que aparentemente hay 5 SPR en la misma línea. Como es esto posible? Haciendo Flickering? NO
Se notaria el parpadeo, el truco está en que en la misma línea horizontal no hay más de 4 SPRs.
Seguro que os estáis preguntando… pues no lo entiendo si hay 5 colores hay 5 SPRs.

Os desvelo el secreto en la siguiente página.

Estos son los 3 SPRs de la Prota.



SPR1 - El pelo Rojo
SPR2 - El vestidito verde
SPR3 - El cuerpo de la Prota

El resultado mezclado de los 3.

Y estos los 2 SPRs del Zombie Verde

SPR1 - 1º color
SPR2 - 2º color

La mezcla de los 2

Si os fijáis en la posición del pelo y del vestido verde no coinciden con la posición donde están en la mezcla, esto es así porque el SPR del pelo va situado justo encima del SPR del vestido y el cuerpo de la prota en medio de los 2, vamos a verlo con el ejemplo de los SPRs con números.

Aquí podéis ver que el SPR1-el Pelo Rojo está situado justo encima del SPR2 que es el vestido Verde, estos 2 SPRs nunca están en la misma línea horizontal mientras que el cuerpo que es el SPR3 está entre medias de los dos. Y aunque en la imagen veas que el SPR4 y SPR5 están separados realmente están juntos pero es para que lo veas bien que hay 5 SPRs, aquí si puedes apreciar que en la misma línea horizontal solo hay 4 SPRs a la vez ya que el SPR1 y el SPR2 no se mezclan en la misma línea horizontal.

Ahora tirar 2 líneas imaginarias en el SPR1 en esa línea horizontal solo coinciden los SPR 1,3,4,5 y si vuelves a tirar 2 líneas imaginarias en el SPR2 en esa línea horizontal solo coinciden los SPRs 2,3,4,5 en cualquiera de los 2 casos siempre hay un máximo de 4 SPRs en la misma línea. Por eso no hay que colocar otro SPR enemigo en la misma línea horizontal porque entre la prota y los 2 enemigos sumarian 6 SPRs en la misma línea.

Pues esto es todo para la primera parte del mundo de los Sprites.
Espero ver vuestros comentarios y vuestras capturas de pantallas, en los BLOG´s o FORO´s del tutorial.

Espero que haya sido de vuestro total agrado y nos vemos en el próximo tutorial. Donde veremos cómo podemos paliar el efecto del 5º SPR, además de incorporar movimientos a los SPRs con la lectura de los cursores y Joystick. Y veremos por primera vez como trabajar con la memoria RAM.

Link para el acceso a los ejemplos o código del tutorial.
http://www.dimensionzgames.com/wp-content/uploads/downloads/2012/03/Tutorial6.rar

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)