VII

Scripting básico…

Este tema es bastante extenso y actúa como continuación del tema 4. Si recordáis, en dicho tema expliqué cómo hacer un script sencillo en el que una persona en el juego te dijera algo. Construimos un script sencillo, aprendimos a escribir los diálogos utilizando los comandos de los textos, compilamos el script y se lo asignamos a un minisprite. Todo eso debería haber quedado más que claro y si seguiste los consejos deberías haber hecho ya varias pruebas con ese modelo de script. No obstante, si crees que no te quedó lo suficientemente claro, recomiendo encarecidamente que se vuelva a repasar el tema 4.

·Personalizando los textos:

Seguimos con el mismo script, un diálogo simple en el que una persona nos dice algo. Sólo aprendimos a hacer saltos de linea, pero a los diálogos se les puede cambiar el color de la fuente, el fondo, el tamaño, podemos hacer que se muestre nuestro nombre o el del rival y muchas cosas más.

Para ello, XSE incorpora una guía en la que podemos encontrar éstos comandos que podemos implementar en nuestros textos. Así que abrimos XSE y vamos a *Ayuda>Guía*. Ahora que estamos en la guía desplegamos *XSE - Comprehensive Scripting Guide* y dentro de *Appendix* abrimos *Message Codes*. Ahí vienen todos los comandos para los mensajes con la estructura "[X]" donde "X" es el nombre del comando. Estos comandos se sustituirán en el juego por algo que queramos mostrar. Hay que fijarse bien para que roms son válidos los comandos, lo cuál se especifica en la guía de XSE.

Por ejemplo, imaginemos que utilizamos Fire Red. Al principio del juego tendremos que ponerle al rival el nombre que queramos, por lo que para mostrar el nombre se utiliza el comando "[rival]". Si queremos, por ejemplo, que al hablar con el rival nos diga "(nombre del rival en verde): Hola, (tu nombre en rojo)" lo haremos de la siguiente manera (para fire red):

= [green_fr][rival]: Hola, [red_fr][player].

Imaginemos que el protagonista se llama Red y el rival Green, pues el texto se vería así:

Green: Hola, Red.

Todo el texto que pongamos detrás del comando para colorearlo saldrá coloreado, por eso "Green: Hola," sale en verde. Pero así queda mal, vamos a colorear sólo los nombres de los

personajes, por lo que habría que añadir el comando "[black_fr]" para colorear el texto de negro. Con lo cuál quedaría así:

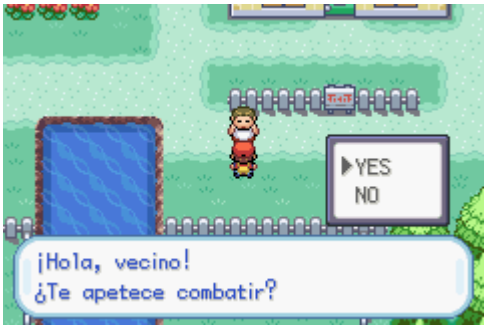
= [green_fr][rival][black_fr]: Hola, [red_fr][player][black_fr].

Green: Hola, Red.

Es cuestión de entender los comandos y cómo funcionan. Lo mejor es comprobar qué comandos existen en la guía de XSE y aprender a usarlos haciendo pruebas con los diálogos.

·Yes/No

Este tipo de script es tan sencillo como el de un diálogo normal, la diferencia es que al acabar el texto aparecerá una caja para responder afirmativa o negativamente (yes o no).



Si antes poníamos *msgbox @(pointer del texto) 0x6* para un mensaje normal, ahora pondremos *msgbox @(pointer del texto)* ***0x5*** y aparecerá la caja. Pero si sólo hacemos ese cambio, respondamos sí o no el script seguirá y no pasará nada. Además de cambiar el tipo de mensaje, hay que comprobar si respondimos sí o no para que pasen diferentes cosas. Aquí tenéis un ejemplo. La primera imagen se corresponde con la respuesta "Yes" y la segunda con la respuesta "No":

Vamos a partir de un script normal al que le hemos cambiado el tipo de mensaje en el *msgbox*:

```
#dynamic 0x800000

#org @comienzo
```

```
lock
faceplayer
msgbox @1 0x5
release
end

#org @1
= ¡Hola, vecino!\n¿Te apetece combatir?
```

Si compilamos ese script, veremos que después de decirnos que si queremos combatir, saldrá la caja para responder. Pero respondamos lo que respondamos no pasará nada. Así que tendremos que poner más cosas en nuestro script. En primer lugar, debéis saber que dependiendo de la respuesta que demos se guardará un valor u otro en la variable *LASTRESULT*. Por lo tanto, para determinar si respondimos sí o no, tendremos que consultárselo a la variable:

```
compare LASTRESULT 0x1
```

Así estamos comprobando si el valor de la variable *LASTRESULT* es 0x1. Ahora haremos que si efectivamente es 0x1, el script salte a otro pointer.

```
compare LASTRESULT 0x1
if 0x1 goto @si
```

Estos dos comandos se pueden traducir para que lo entendáis:

Comprobar si el valor de *LASTRESULT* es 0x1
Si lo es ir a @si

Como podéis ver es mucho más sencillo de lo que parece. Ahora también podríamos poner *if 0x0 goto @no*, pero nos lo ahorraremos por una razón muy sencilla. Si el script no salta al pointer @si, deducimos que hemos respondido "no", por lo tanto lo que haremos es seguir en el mismo pointer poniendo lo que nos dirá si contestamos "no":

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
msgbox @1 0x5
compare LASTRESULT 0x1
if 0x1 goto @si
msgbox @2 0x6
release
end

#org @1
= ¡Hola, vecino!\n¿Te apetece combatir?
```

```
#org @2
= Sabía que dirías que no.\nSoy demasiado fuerte para ti.
```

¿Veis? Si respondimos no, el script no saltará al pointer @si y seguirá leyendo. Y lo que leerá será el *msgbox* con el texto que se mostrará si nuestra respuesta fue "no". Ahora vamos a scriptear el pointer @si:

```
#org @si
msgbox @3 0x6
release
end

#org @3
= ¿Sí? Pues olvídalo. No pienso gastar\nenergías en derrotarte.
```

Por lo tanto, el script completo quedaría de la siguiente manera:

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
msgbox @1 0x5
compare LASTRESULT 0x1
if 0x1 goto @si
msgbox @2 0x6
release
end

#org @si
msgbox @3 0x6
release
end

#org @1
= ¡Hola, vecino!\n¿Te apetece combatir?

#org @2
= Sabía que dirías que no.\nSoy demasiado fuerte para ti.

#org @3
= ¿Sí? Pues olvídalo. No pienso gastar\nenergías en derrotarte.
```

Si os dáis cuenta, en el pointer @si no he utilizado ni *lock* ni *faceplayer*. La razón es porque ya lo hemos puesto en @comienzo y el posible salto a @si (en caso de responder afirmativamente) está después, por lo que siempre se van a leer y a ejecutar ambos comandos. Por eso no es necesario volver a ponerlos otra vez. El *release*, en cambio, hay que ponerlo ya que en caso de que el script saltara a @si no se leería el *release* de @comienzo.

·Flags

Las flags son interruptores. Cada flag puede tener dos estados distintos: Activada y Desactivada. Una flag es como un interruptor de una bombilla. Si el interruptor está activado se encenderá la bombilla y si no lo está la bombilla no se encenderá. Si la flag está activada pasará una cosa y si no lo está no pasará nada o pasará otra cosa distinta. Pero las flags hay que explicarlas por partes porque tienen tres funciones distintas a las que les he dado distintos nombres:

1. **Condicionales:** Se usan en los scripts para que pase una cosa u otra dependiendo de si está o no activada.
2. **Especiales:** Sirven para activar o desactivar una función especial del juego.
3. **Flags de aparición:** Sirven para hacer desaparecer o aparecer un minisprite en un momento determinado y de manera indefinida.

Para el uso de las flags se utilizarán, por otro lado, tres comandos básicos:

1. **Setflag:** Activa la flag indicada con la estructura setflag *0x(flag).*
2. **checkflag:** Comprueba si la flag está o no activada con la estructura checkflag *0x(flag).*
3. **clearflag:** Desactiva la flag indicada con la estructura clearflag *0x(flag).*

Otra cosa muy importante que debéis saber es qué flags podéis utilizar. No podéis usar las que queráis, sino las que estén disponibles:

Desde la **0x200** hasta la **0x29F**
Desde la **0x500** hasta la **0x79F**
Desde la **0x1000** hasta la **0x109F**

Las flags, como era de esperar, están en hexadecimal. Así que tened cuidado con eso y recordad que después de la 0x209 no va la 0x210, sino la 0x20A (y así con todas). También es importante aclarar que las flags, por defecto, están desactivadas.

Flags condicionales:

Este es uno de los usos de las flags. Se utilizará para comprobar sí algo ya pasó y, entonces, que pase otra cosa. Vamos a verlo con dos ejemplos. Uno en el mismo script y otro en dos scripts diferentes. **Vamos con el primer ejemplo**.

Imaginemos que queremos hablar con una persona que nos diga algo. Pero queremos que sólo nos lo diga una vez y si volvemos a hablar con esa persona, que nos diga otra cosa. Es decir:

Primera vez que hablamos: "El Profesor Oak quiere que vayas a verle"
Si volvemos a hablar con él: "No te retrases o se enfadará"

Construimos nuestro script normal con el primer texto:

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
msgbox @1 0x6
release
end

#org @1
= El Profesor Oak quiere que vayas a\nverle.
```

Una vez que nos haya dicho eso, ya no queremos que vuelva a repetirlo, por lo que haremos uso de una flag. Al final del script activaremos una flag. Como todavía no hemos usado ninguna, utilizaremos la 0x200 que está libre y la activaremos con el comando *setflag 0x200*. Quedaría algo así:

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
msgbox @1 0x6
setflag 0x200
release
end

#org @1
= El Profesor Oak quiere que vayas a\nverle.
```

Pero si ejecutamos ese script, seguirá diciéndonos lo mismo una y otra vez. La razón es que hemos activado la flag, pero no hemos hecho la comprobación de que está activada y hemos saltado a otro pointer. Esto es igual que en el *yes/no*. Y comprobaremos si la flag está activada de esta manera:

```
checkflag 0x200
if 0x1 goto @despues
```

Esto se puede traducir como:

Comprobar si la flag 0x200 está activada
Si lo está saltar a @despues

También lo podemos hacer al revés. Es decir, estamos comprobando si está activada, pero también podríamos comprobar que no está activada. Esto se hace así:

```
checkflag 0x200
if 0x0 goto @despues
```

Que vendría significando lo siguiente:

Comprobar si la flag 0x200 está activada
Si no lo está saltar a @despues

Estos comandos debemos ponerlos después de los comandos que queremos que se repitan y antes de los que no. En nuestro caso queremos que el *msgbox* no se repita, pero que si lo hagan el *lock* y el *faceplayer* para no tener que ponerlos de nuevo. Así que quedaría así:

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
checkflag 0x200
if 0x1 goto @despues
msgbox @1 0x6
setflag 0x200
release
end

#org @1
= El Profesor Oak quiere que vayas a\nverle.
```

Ahora lo único que tenemos que scriptear es el pointer @despues en el que simplemente pondremos el segundo texto:

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
checkflag 0x200
if 0x1 goto @despues
msgbox @1 0x6
setflag 0x200
release
end

#org @despues
msgbox @2 0x6
release
```

```
end

#org @1
= El Profesor Oak quiere que vayas a\nverle.

#org @2
= No te retrases o se enfadará.
```

Como podéis ver es muy similar al script de *yes/no*. Y esto lo podéis utilizar en cualquier tipo de script siempre que queráis poner una condición para que pase algo. Es importante recordar que si hemos utilizado la flag 0x200, ésta ya estará ocupada y tendremos que utilizar otra distinta si lo necesitamos, ya que desactivar la flag 0x200 puede afectar al script en el que la hayamos usado. Por supuesto se puede utilizar para algunos eventos y así ahorrarnos flags, pero eso ya lo aprenderéis y es mejor dejarlo por ahora.

Pero no lo vamos a dejar aquí porque nos queda otro ejemplo. **Las flags son globales**, es decir, no se activan o desactivan sólo dentro de un script, sino que lo hacen para todo el juego. Es decir, si en un script al principio del juego activas una flag, llegarás al final del juego y esa flag seguirá activada para todos los scripts. Para aprender y demostrar esto, **vamos con el segundo ejemplo**.

Imaginemos que queremos que una persona nos diga "¿Eres de por aquí?" la primera vez que hablamos con él y que cuando hablemos con otra persona distinta que nos diga "Ese chico de allí enfrente es mi hermano.", volvamos a hablar con el primero y nos diga "¡Ah, sí! Tu eres de Pueblo Paleta".

Para hacer esto tendremos que hacer dos scripts, uno para cada uno de los minisprites. Así que vamos con el primero, que sería así:

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
checkflag 0x201
if 0x1 goto @despues
msgbox @1 0x6
release
end

#org @despues
msgbox @2 0x6
release
end

#org @1
= ¿Eres de por aquí?
```

```
#org @2
= ¡Ah, sí! Tu eres de Pueblo Paleta.
```

Ahora hemos utilizado la flag 0x201 porque la 0x200 la utilizamos en el anterior script. Y el script es igual que el del anterior ejemplo, si os dais cuenta, pero le falta el *setflag*. Esto se debe a que queremos que cuando la flag 0x201 esté activada, nos diga otra cosa. Pero queremos que eso pase después de hablar con otra persona, por lo que activaremos la flag en el script de esa otra persona. Quedaría así:

```
#org @comienzo
lock
faceplayer
msgbox @1 0x6
setflag 0x201
release
end

#org @1
= Ese chico de allí enfrente es mi\nhermano.
```

Ya están los dos scripts. Si los compilamos y los probamos, veremos que hasta que no hablemos con la segunda persona, la primera no nos dirá el segundo texto. Es decir, en el script de la primera persona se comprueba si la flag está activada, pero no se activa hasta hablar con la segunda persona, por lo que hasta ese momento no nos dirá el segundo diálogo.

Flags especiales:

Las flags especiales funcionan igual que las anteriores. Se activan, desactivan y comprueban igual. La diferencia es que si activas o desactivas una de estas flags, estarás activando o desactivando un sistema del juego. Por ejemplo, si en Fire Red activas la flag 0x820 desbloquearás la primera medalla. Si después la desactivas volverás a bloquear dicha medalla. **Revisa el apéndice para ver la lista de flags especiales**. Pero aquí también se pueden comprobar las flags, como he dicho antes. Por ejemplo, si queremos que algo no pase hasta que tengamos la primera medalla podemos usar el *checkflag 0x820* y comprobar de esa forma si la tenemos o no. Un ejemplo de esto es la liga pokémon, cuando comprueban antes de pasar si tenemos todas las medallas. Lo que hace ese script es comprobar si las flags que activan las medallas están activadas mediante el comando, precisamente, *checkflag*.

Flags de aparición:

Vuelven a funcionar igual que las condicionales y se pueden utilizar las mismas flags. Se pueden activar, desactivar y comprobar. Pero en esta ocasión utilizaremos las flags para hacer aparecer o desaparecer a un personaje en el juego.

Si recordáis, las flags están desactivadas por defecto. Para que un personaje desaparezca hay

que activar la flag que le asignemos. Lo que quiere decir que si queremos que esté oculto hasta que queramos que aparezca, tendremos que activar la flag antes de llegar a donde se encuentra el personaje. Es decir, si ponemos un personaje en Pueblo Paleta y queremos que tras hablar con él desaparezca, simplemente le asignamos una flag y la activamos al final del script. Pero si por el contrario queremos que antes de que ese personaje sea visible haya que ir al laboratorio de Oak, tendremos que activar la flag antes de llegar al mapa de Pueblo Paleta.

Explicaré las dos cosas por separado. **Vamos con el primer ejemplo**.

Imaginemos que como en el juego original, comenzamos nuestra aventura en la habitación de la casa del protagonista. En pueblo Paleta añadimos un minisprite con Advance Map o editamos uno existente con un script sencillo en el que nos diga algo. Tras decirnos ese algo, el personaje desaparecerá. Para ello vamos a escoger una flag que no esté usada. Por ejemplo, la 0x200 (si no la hemos usado ya).

El personaje nos dirá "El profesor Oak quiere verte" y después desaparecerá. Para ello, construiremos nuestro script y, al final, activaremos la flag que elegimos:

```
#org @comienzo
lock
faceplayer
msgbox @1 0x6
setflag 0x200
release
end

#org @1
= El profesor Oak quiere verte.
```

Si hablamos con él, nos dirá el texto pero nunca desaparecerá. Eso se debe a que hay que asignarle la flag al personaje. Para ello lo seleccionamos en Advance Map y vamos a "People ID". Si el personaje lo hemos añadido nosotros, debería poner "0000" si no lo hemos editado. Ahí es donde debemos poner nuestra flag para asignársela. Pero por defecto hay **cuatro caracteres (0000)**, por lo que nosotros tenemos que poner cuatro también. Es decir, para asignarle la flag 0x200, pondremos "0200", es decir, se le añade un 0 delante para completar los cuatro valores. Guardamos y ya hemos asignado nuestra flag.

Si ahora hablamos con el personaje, nos dirá el texto y si entramos en otro mapa y volvemos o simplemente nos alejamos lo suficiente del personaje, veremos que al volver ya no estará ahí. Es una manera de desaparecer natural, es decir, le pierdes de vista y cuando vuelves ya no está. Pero hay veces que necesitaremos que desaparezca justamente tras hablar con él. Se puede utilizar el comando *fadescreen* para poner la pantalla en negro y que al volverse a poner bien el personaje ya no esté. También podemos aplicarle un movimiento al personaje para que salga de la pantalla y entonces hacerle desaparecer (como si se marchara). Para ésto utilizaremos el comando ***hidesprite*** con la estructura *hidesprite 0x(número del minisprite)*.

Si utilizamos éste comando, el personaje desaparecerá instantáneamente, pero no vas a hacer que desaparezca de la pantalla de buenas a primeras, no queda bien, por eso decía que podemos volver la pantalla negra o aplicar un movimiento al personaje y cuando no esté en nuestro campo visual, aplicar este comando para hacerlo desaparecer de una forma más profesional. El número del minisprite se puede ver seleccionándolo en Advance Map y mirando el número que aparece en "No. de Gente".

Es importante recordar que el comando *hidesprite* hace desaparecer a nuestro personaje, pero si no hemos activado una flag asignada al mismo, volverá a aparecer nada más movernos.

Ahora vamos con el segundo ejemplo.

Imaginemos que empezamos el juego en nuestra habitación. Queremos ir al laboratorio a hablar con un personaje y que después de haber hablado con ese personaje, aparezca otro que estaba oculto en Pueblo Paleta.

Esto lo mejor es hacerlo mediante script de nivel, pero como todavía no hemos llegado a eso utilizaremos como hasta ahora scripts asignados a minisprites. Pondremos un personaje en nuestra casa que al hablar con él nos diga cualquier cosa, lo que queráis, y al final del script se active una flag, por ejemplo y si no la hemos usado ya, la 0x201. Luego ponemos otro personaje en el laboratorio igual, que nos diga lo que sea, pero esta vez no activaremos la flag, sino que la desactivaremos con el comando *clearflag 0x201*.

Ahora si hablamos con el de casa se activa la flag y si hablamos con el del laboratorio se desactiva. Pues ahora ponemos a nuestro tercer personaje y lo ponemos en Pueblo Paleta asignándole la flag 0x201 y poniéndole otro script en el que diga cualquier cosa (esta vez sin activar o desactivar flags en él).

Finalmente probáis los scripts y veréis que el personaje de Pueblo Paleta aparece y desaparece dependiendo de con quién habléis. Eso os debería servir para entender esta función de las flags. No obstante, recomiendo que hagáis vuestras propias pruebas.

·Checkgender

Como todos sabéis, los juegos originales nos ofrecen la posibilidad de elegir entre el personaje masculino o el femenino. La función *checkgender* nos dará la posibilidad de comprobar en el script si estamos jugando con la chica o con el chico. Por ejemplo, si elegimos al chico sería raro que alguien nos dijera "Eres una chica muy hermosa." ¿No? Este problema se nos presentará principalmente a la hora de utilizar adjetivos (cansado/cansada, aburrido/aburrida, ganador/ganadora...).

Hasta ahora hemos utilizado el comando *goto* para ir a otra parte del script (*if 0x1 goto*

@(pointer)), pero aprovechando este apartado aprenderemos a utilizar, en vez de este, el comando *call*. La razón es sencillamente que funciona igual que el otro pero nos da la opción de regresar al script con el comando *return* y seguir leyendo desde donde se produjo el *call*. Así aprenderemos a ahorrar espacio en determinadas ocasiones, como esta. Primero aprenderemos a hacerlo con *goto* y, depués y para que sea más fácil de comprender, lo haremos con *call*. Si entendiste el funcionamiento del yes/no esto es pan comido:

```
checkgender
compare LASTRESULT (0x0 para comprobar si es chico. 0x1 para comprobar si es chica)
if 0x1 goto @(pointer)
```

Por ejemplo. Si queremos que al hablar con alguien nos diga "¡Eres el mejor entrenador del mundo!" o "¡Eres la mejor entrenadora del mundo!", lo podemos hacer de la siguiente manera:

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
checkgender
compare LASTRESULT 0x1
if 0x1 goto @chica
msgbox @1 0x6
release
end

#org @chica
msgbox @2 0x6
release
end

#org @1
= ¡Eres el mejor entrenador del\nmundo!

#org @2
= ¡Eres la mejor entrenadora del\nmundo!
```

Con el *checkgender* estamos comprobando si es chica y, en caso afirmativo, nos llevará al pointer *@chica*. De lo contrario, seguirá leyendo el script. Esto es sencillo, pues ya lo hemos visto antes en otros tipos de scripts. Esto no requiere mucha más explicación, pero vamos a aprovechar como se ha dicho antes para aplicar *call* en vez de *goto*:

```
#dynamic 0x800000

#org @comienzo
lock
faceplayer
```

```
checkgender
compare LASTRESULT 0x1
if 0x0 call @chico
if 0x1 call @chica
msgbox @3 0x6
release
end

#org @chico
msgbox @1 0x6
return

#org @chica
msgbox @2 0x6
return

#org @1
= ¡Eres el mejor entrenador del\nmundo!

#org @2
= ¡Eres la mejor entrenadora del\nmundo!

#org @3
= Lástima que no te presentes a\nla liga Pokémon.
```

Comprobará si somos chico o chica y nos llevará, según el caso, al pointer *@chico* si somos chico o *@chica* si somos chica. Pero hemos utilizado *call*, por lo que nos dará la opción de volver. Si os fijáis, hemos puesto *return* al final de ambos pointers, lo que hará que el script se siga leyendo desde el comando *call*. Esto con un *goto* no sería posible. Tampoco es necesario volver si no hace falta, pero es importante que comprendáis esta función para el futuro. Deberíais compilar este script y probarlo para entender mejor el funcionamiento.

·Applymovement

Se trata de un comando muy importante que nos permitirá aplicarle movimiento a un minisprite o a la propia cámara del juego. En principio vamos a centrarnos en aplicar movimiento a un minisprite. Funciona un poco como el comando *msgbox*. Ahora veréis por qué. El comando tiene la estructura ***applymovement 0x(número de minisprite) @(pointer del movimiento)***. Como en el *msgbox* poníamos el texto a parte, aquí pondremos los movimientos a parte también, en otro pointer, de la siguiente manera:

```
#org @(pointer)
(movimientos)
#raw 0xFE
```

Ese #raw *0xFE* lo que hace es finalizar los movimientos. Es imprescindible ponerlo al final o se bloqueará el juego. **La lista de movimientos podéis encontrarla en el apéndice**. Vamos a

hacer un ejemplo en el que al hablar con un mini se mueva dos pasos hacia arriba. Para saber cuál es el número del minisprite recordad que tenéis que seleccionarlo en advance map y mirarlo en *No. de Gente*. En este caso supongamos que es el número 2, por ejemplo.

```
#org @comienzo
lock
applymovement 0x2 @mov
waitmovement 0x0
release
end

#org @mov
#raw 0x11
#raw 0x11
#raw 0xFE
```

En este caso, al interactuar con el mini, se moverá hacia arriba. Si hay algo arriba, lo atravesará, incluso nos atravesará a nosotros. Por eso hay que tener cuidado al realizar los applymovements y estar seguro de que no va a atravesar nada ni a nadie. Lo más normal, por ello, es aplicar este tipo de comandos en scripts de gatillo o nivel, en los que sabemos seguro cuál va a ser la posición de cada personaje. Pero eso ya lo aprenderéis más adelante.

Os habréis fijado también que hemos incluido el comando *waitmovement* al script. Este comando sirve para que el script espere a que se ejecute el movimiento antes de continuar. Si no lo ponemos, no se producirá el movimiento. Tiene la estructura *waitmovement 0x(número de mini)*, pero podéis poner directamente *waitmovement 0x0* para esperar a todos los movimientos. De esa forma no hay que especificar el número de mini. Digamos, pues, que es universal para todos los minisprites.