

# XI

## Scripts de gatillo y de nivel

Hasta ahora hemos visto dos tipos de scripts. Estos son los scripts asignados a la gente o asignados a postes. En ambos, había que ir hasta ellos y pulsar el botón A para interactuar y ejecutar el script. Pero hay otros dos tipos de scripts más y son los de gatillo y los de nivel. A diferencia de los otros, estos scripts son invisibles y automáticos. Son invisibles porque no se pueden ver en el mapa. En los dos anteriores veíamos perfectamente a la gente o los tiles que pudieran tener asociados un script de poste (un cartel, por ejemplo) a excepción de los objetos ocultos, claro, pero en este caso no sabremos nunca de su existencia hasta que no se ejecuten, es decir, antes no los veremos y, por tanto, no sabremos que ahí hay un script. Por otro lado, son automáticos porque el jugador no va a ellos, sino que se ejecutan automáticamente cuando se den las circunstancias adecuadas.

El script de gatillo se ejecutará al pisar una cierta parte del mapa. El script de nivel, por su parte, se ejecutará al entrar en un mapa o, estando ya en el mapa, cuando se den las circunstancias adecuadas. Ambos tipos de scripts son configurados mediante variables.

### •Variables

Lo primero es comprender el funcionamiento de las variables. En ambos tipos de scripts jugarán el mismo papel, por lo que no será demasiado complicado.

Lo más importante es saber que las variables son como las flags. Llegados a este punto, deberías saber cómo funcionaban éstas. Si no lo sabes o no lo recuerdas, deberías leer esa parte del manual antes de seguir con esto. Recordamos que las flags podían estar activadas o desactivas y, en función de cómo estuvieran, pasaría algo o no pasaría, o pasaría una cosa u otra. Las variables son iguales, pero la diferencia es que las variables no están activadas o desactivadas, sino que guardan un valor. **Utilizaremos desde la 0x7FFF hacia atrás.** Las variables guardan un valor de 2 bytes, por lo que pueden ir desde el 0x0000 hasta el 0xFFFF. Un total de 65536 valores distintos puede contener la variable. Como podéis ver, dan mucho más juego que las flags. **Es importante saber que las variables tienen por defecto el valor 0.**

Un script de gatillo o de nivel se ejecutará cuando la variable especificada tenga el valor especificado. Por ejemplo, sabemos que un script de gatillo se ejecuta cuando se pisa, pero siempre que se cumplan unas condiciones. Estas condiciones serán que una determinada

variable tenga un valor concreto. Para continuar con el ejemplo vamos a escoger la variable 0x7FFF y vamos a determinar que el script se ejecutará cuando esta variable tenga el valor 0x1 (esto se hace a la hora de configurar el script en Advance Map, que lo veremos después, de momento es únicamente necesario conocer la teoría). Como el valor por defecto de la variable, si no lo hemos cambiado mediante un script anterior, es 0x0, cuando pisemos el script, éste determinará que el valor de la variable 0x7FFF es 0x0 y no 0x1, por lo tanto, el script no se ejecutará. En el momento que se ejecute otro script de cualquier tipo y se le dé en él el valor 0x1 a nuestra variable, si volvemos a pisar el script, esta vez la variable sí tendrá el valor especificado en la configuración, por lo que esta vez sí se ejecutará.

Ese script de gatillo se seguirá ejecutando cada vez que lo pisemos si el valor de la variable es el que configuramos. Por lo que, si queremos que deje de ejecutarse, le daremos un valor nuevo a la variable y distinto al que configuramos. Por ejemplo, si escogimos el valor 0x1, ahora podemos darle el valor 0x2, entonces ya no se ejecutará más. La clave para que deje de ejecutarse es que sea cualquier valor excepto el que escogimos en la configuración, naturalmente. Y esto lo vamos a hacer cambiando el valor de la variable en el propio script de gatillo o en otro script distinto. Dependerá de las circunstancias. Lo comprenderéis fácilmente con dos ejemplos.

Imaginemos que al pisar nuestro script de gatillo se nos acercará una persona, nos dará un objeto y se irá. Si volvemos a pisar el script, volverá a venir la persona a darnos de nuevo el objeto. Pero no queremos que pase eso. Entonces en el propio script de gatillo, cambiaremos el valor de la variable para que no se vuelva a repetir.

Pero ahora imaginemos que ponemos el script de gatillo en la salida de un pueblo para que no podamos salir de él antes de hacer algo. No tendría sentido que se desactivase el script en el propio script, ya que te diría que no puedes irte, pero si lo intentas de nuevo, ya sí podrás. Es completamente inverosímil. Simplemente haremos eso que teníamos que hacer antes de salir, por ejemplo, hablar con una persona. Y en el script de esa persona, cambiamos el valor de la variable, entonces si volvemos a pisar el script de gatillo, ya estará desactivado.

No obstante, las variables no sólo sirven para configurar scripts de gatillo y nivel, sino también para utilizarlas como condicionales en un script, al igual que las flags. Vamos a ver un ejemplo para comprenderlo mejor. Imaginemos que queremos que una persona nos de algo, pero para que nos lo de, tenemos que hablar antes con otras cinco personas. Perfectamente podemos usar una flag distinta para cada una de esas personas y, cuando hablemos con ella, se active la que le corresponda. Luego en el script de la persona que nos dará el objeto, comprobamos si las cinco flags están activadas ¿Pero qué pasa? Lo primero es que hay que gastar cinco flags distintas para hacer esto. Y lo segundo es que tendremos que comprobar en el script el estado de las cinco flags, cuando podemos tan solo comprobar el de una única variable. En vez de activar una flag por cada persona, podemos escoger una variable y, cuando hablemos con esas personas, se suma un valor a la variable, por lo que, cuando hablemos con todas las personas, la variable tendrá el valor 0x5, que es lo único que habrá que comprobar. Es sólo un ejemplo práctico para que veáis que se pueden usar para otras muchas cosas.

Vamos a ver los comandos más importantes que utilizaremos para tratar con las variables.

**Setvar:** Sirve para darle un valor concreto a una variable mediante *setvar 0x(variable) 0x(valor)*. Por ejemplo, si queremos darle el valor 0x8 a la variable 0x7FFF utilizaremos *setvar 0x7FFF 0x8*.

**Addvar:** Lo utilizaremos para sumar un valor determinado al valor que ya tiene la variable mediante *addvar 0x(variable) 0x(valor a sumar)*. Por ejemplo, si la variable tiene el valor 0x3 y ponemos *addvar 0x7FFF 0x2*, se sumarán 0x3 valores a la variable, por lo que ahora tendrá el valor 0x5.

**Subvar:** Es igual que el anterior, pero en vez de sumar un valor al valor de la variable, se lo resta. Por ejemplo, si nuestra variable tiene el valor 0x3 y ponemos *subvar 0x7FFF 0x2*, le restaremos 0x2 valores a nuestra variable, por lo que se quedará con el valor 0x1.

**Copyvar:** Este comando nos permitirá copiar el valor de una variable a otra variable. Es muy práctico, sobre todo, cuando entra en juego la variable 0x800D (llamada "LASTRESULT"). En la variable LASTRESULT se guardan continuamente valores, por lo que utilizar el comando *copyvar* nos ayudará a salvar un determinado valor guardado en LASTRESULT en una variable segura. Como siempre, es un ejemplo, se pueden hacer infinidad de cosas. El funcionamiento es *copyvar 0x(variable de destino) 0x(variable en la que tenemos el valor)*. Es decir, si queremos guardar el valor de LASTRESULT en la variable 0x7FFF, pondremos *copyvar 0x7FFF LASTRESULT*, o si en vez de poner "LASTRESULT" queremos poner la propia variable que es 0x800D, lo pondremos como *copyvar 0x7FFF 0x800D*.

**Compare:** Sirve para comprobar si una variable tiene un determinado valor. Se utiliza *compare 0x(variable) 0x(valor)*. Por ejemplo, si queremos comprobar si la variable 0x7FFF tiene el valor 0x1, pondremos *compare 0x7FFF 0x1*.

**Comparevars:** Compara el valor de dos variables. Se compara el valor de la variable A respecto a la variable B. Se utiliza *comparevars 0x(variable B) 0x(variable A)*.

En los supuestos de *compare* y *comparevars*, comprobaremos el resultado con los siguientes valores:

- **0x0:** Menor que (valor).
- **0x1:** Igual que (valor).
- **0x2:** Mayor que (valor).
- **0x3:** Igual o menor que (valor).
- **0x4:** Igual o mayor que (valor).
- **0x5:** Diferente de (valor).

Lo comprenderemos fácilmente con un ejemplo. Pero antes hagamos memoria y recordemos las flags. Lo que hacíamos era comprobar mediante el comando *if* si la flag estaba o no activada. Por ejemplo, podíamos poner esto:

```
Checkflag 0x200  
If 0x1 goto @asdf
```

Que era algo así:

```
Comprueba el estado de la flag 0x200  
Si está activada, va a @asdf
```

Esto es un poco más complejo, pero muy similar. Utilizaremos *if 0x(valor) goto @(pointer)*, donde el valor será uno de los que vimos arriba. Imaginemos lo más simple, que queremos que en un script vayamos a un pointer concreto si la variable 0x7FFF tiene el valor 0x2:

```
Compare 0x7FFF 0x2  
If 0x1 goto @asdf
```

Comparará el valor que tiene la variable 0x7FFF con respecto al valor que nosotros queramos. En este caso, el valor 0x2. Y como en el *if* pusimos 0x1 ("igual que"), nos llevará al pointer @asdf siempre que el valor de la variable sea 0x2.

Si por ejemplo hubiéramos querido que nos llevara al pointer @asdf siempre que el valor de la variable fuera distinto al valor que especifiquemos, utilizaríamos el 0x5:


```
Compare 0x7FFF 0x2  
If 0x5 goto @asdf
```

En este supuesto, siempre que el valor de la variable sea diferente a 0x2, irá a @asdf. Dependiendo de cómo funcione el script, tendremos que poner que sea mayor, menor, igual... Simplemente poner el valor que proceda.

Recordad que estamos poniendo un *goto* en el *if*, pero también se puede poner un *call*. Recordad que la diferencia es que con el *call*, se podrá utilizar un *return* para volver al punto en el que se utilizó el *call*. Pero eso ya se explicó en temas anteriores.

Con esto podemos dar por concluido el apartado de variables. Claro que las variables son la base, ya que los scripts de gatillo y nivel son configurados mediante ellas. Si no se entiende las variables, no es aconsejable que pases al siguiente punto. En tal caso, vuelve a leerlo tranquilamente comprendiendo cada cosa que se explica. Si te supone cualquier problema, coge un rom para experimentar y/o investigar scripts originales. También puedes pedir ayuda en algún foro de Rom Hacking. Si por el contrario lo has comprendido todo, enhorabuena, puedes continuar con el manual.

## •Scripts de gatillo

Un script de gatillo es un script que se ejecuta cuando es pisado por el jugador. En Advance Map es representado así: 

Es un script normal y corriente como el que le asignamos a cualquier overworld del juego. Claro que en este caso hay que olvidarse de algunos comandos como el *lock* o el *faceplayer*, ya que no tendrán ningún tipo de efecto. Recordemos que el *lock*, detiene el movimiento del mini con el que hablas. Tratándose de un script de gatillo, lo único que podemos hacer si necesitamos detener el movimiento de los minis, es utilizar el comando *lockall* para aplicarlo a todos los minis. Para reactivar el movimiento usaremos *releaseall*.

Hay mucha gente que, para que un script de gatillo no se repita, ponen un *checkflag* que lleva a un pointer donde sólo hay un end. De esta manera, la pisar el gatillo no pasará nada. Pero esto es una solución tan chapucera como incoherente. Un script de gatillo (igual que los de nivel) siempre se desactiva mediante variable.

El funcionamiento es sencillo. El script de gatillo se ejecutará al pisarse siempre que la variable que pongamos tenga el valor que también pongamos en la configuración del script en Advance Map. Para configurarlo, pinchamos en el gatillo que queramos configurar y tendremos que ver algo así:

Pos(X/Y):	0007	0009
Desconocido:	0000	
Var Number:	0000	
Var Value:	0000	
Desconocido:	00	00
Script offset:	\$000000	

En *Desconocido* pondremos siempre 0003. En *Var Number* pondremos la variable que queramos usar para el script. En *Var Value* pondremos el valor (en hexadecimal) que tendrá que tener la variable para que se ejecute el script. Por último, *Script offset* es el offset del script.

Si ponemos la variable 7FFF y el valor 0001, el script sólo se ejecutará cuando el valor de la variable 0x7FFF tenga el valor 0x1. Entonces, para desactivar el script, simplemente hay que dar otro valor a la variable cuando corresponda y ya está.

Es importante tener en cuenta que no es necesario muchas veces usar una variable para cada script, porque, por ejemplo, puedes poner un script que no te deje salir del primer pueblo hasta que hagas algo. Pero luego esa misma variable, puede ser utilizada para otro script más adelante. Aunque esto es una de esas cosas que se van haciendo conforme se gana más

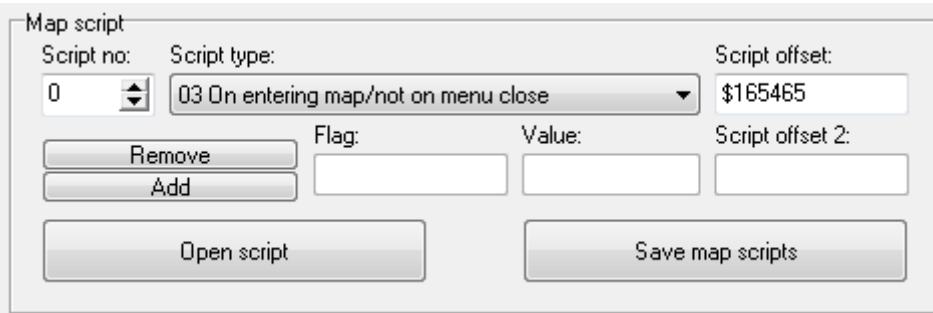
experiencia en el manejo de las variables. Pero es algo a tener en cuenta porque te permite ahorrar muchas variables.

Aconsejo, además, **crear un archivo de texto** junto a nuestro hack **en el que hagamos un listado de las flags y variables** que vayamos usando y así evitaremos problemas de no saber qué flag/variable usar o usar alguna que no recordemos que ya hemos usado antes.

## •Scripts de nivel

Esto es un poco más complejo que el script de gatillo. Sencillamente porque hay 7 tipos de scripts de nivel. Aun así, sólo se conoce el funcionamiento de 6 de ellos. Entonces, tenemos que saber para qué sirve cada tipo de script y cuándo debemos usarlos.

Para añadir un script de nivel, vamos al mapa en el que queramos ponerlo y nos dirigimos a la pestaña *Ver Cabezal*. Una vez en el cabezal, nos fijaremos en la parte que dice *Map script*. En esa parte trabajaremos a la hora de insertar un script de nivel. Estas son las opciones:



The screenshot shows a 'Map script' window with the following elements:

- Script no:** A dropdown menu showing '0'.
- Script type:** A dropdown menu showing '03 On entering map/not on menu close'.
- Script offset:** A text input field containing '\$165465'.
- Buttons:** 'Remove' and 'Add' buttons are located below the 'Script type' field.
- Fields:** 'Flag:', 'Value:', and 'Script offset 2:' are empty text input fields.
- Bottom Buttons:** 'Open script' and 'Save map scripts' are large buttons at the bottom.

- **Script no:** Número de script seleccionado. Es un simple navegador para ver la configuración de cada uno de nuestros scripts de nivel.
- **Script type:** El tipo que tendrá nuestro script de nivel. Dependiendo de lo que queramos hacer, tendrá un tipo u otro (explicado más adelante).
- **Remove:** Elimina el script de nivel seleccionado.
- **Add:** Añade un nuevo script de nivel.
- **Script offset:** Es el offset de nuestro script de nivel. En los scripts de nivel de tipo 01, 03, 05 y 07; la estructura de los scripts de nivel apunta directamente al script, por lo que pondremos el offset del script en esa caja. Por el contrario, en los de tipo 02 y 04, ese offset será el de una subestructura donde se encuentra la configuración, por lo que no tendremos que poner nada ahí (lo crea automáticamente Advance Map).
- **Script offset 2:** En los casos de scripts de nivel de tipo 02 y 04, pondremos el offset ahí, ya que es el que apunta al script (dentro de la subestructura).
- **Flag:** Se trata de la variable que le asignaremos a nuestro script de nivel en caso de que sea de tipo 02 o 04. Aunque ponga flag, será siempre una variable.

- **Value:** Se trata del valor que tendrá que tener la variable para que se ejecute el script de nivel. Igualmente, será en los casos de que el script sea de tipo 02 o 04.
- **Open script:** Abre el script con editor que tengamos vinculado (XSE normalmente).
- **Save map scripts:** Los scripts de nivel están formados por una estructura y, en algunos casos, subestructuras. Por lo que le daremos ahí para que Advance Map cree automáticamente el mapa.

**Importante:** Lo que he explicado respecto a la estructura y subestructura de los scripts de gatillo, realmente no hace falta saberlo, así que no os preocupéis si os parece lioso. Advance Map lo hace automáticamente, así que lo único necesario será saber configurar el script en el propio Advance Map.

Ahora veremos los tipos de scripts, para qué sirven y en qué casos debemos usarlos. Si miramos en el desplegable de los tipos de scripts de nivel en Advance Map, veremos que hay un número y el nombre del script. Generalmente el nombre se omite y basta con poner el número para identificar el script. Por eso, sólo nombraré el número a la hora de referirme a un tipo de script de nivel. Y estos son los tipos:

- **01:** Se trata de un tipo de script para hacer *setmaptiles*. El *setmaptile* es un comando de XSE que sirve para cambiar cualquier bloque del mapa por otro. Por ejemplo, si hay un árbol en el mapa pero en un momento determinado queremos que desaparezca, utilizaremos el comando *setmaptile* para cambiar esos bloques. Este tipo de script se ejecuta cuando entras al mapa. No se puede poner una condición como en el script de gatillo, si pones un script de este tipo, se ejecutará permanentemente. El secreto está en poner una condición dentro del propio script (un *checkflag* o un *compare*) para que se ejecute cuando queramos.
- **02:** Este es igual que un script de gatillo, pero se ejecuta al entrar al mapa o, estando ya en el mapa, cuando la variable tenga el valor indicado. En *flag* pondremos la variable de la que queramos que dependa el script y en *value*, el valor que debe tener la variable para que se ejecute. Es prácticamente igual que un script de gatillo. Pero este es el tipo de script de nivel más complejo, por lo que lo explicaremos después más detalladamente.
- **03:** En script de tipo 02 se utiliza para hacer algo similar al script de gatillo (puedes poner un *applymovement*, un *msgbox*...), pero el de tipo de 03 no desencadena eventos directos como cuando se acerca una persona a hablarnos. Este tipo de script se suele utilizar mucho en el rom original para mover minisprites en determinados casos o para ese tipo de cosas. Lo mejor es ver los scripts del juego original para ver en qué se suelen utilizar. Como el 01, este es un tipo de script que se ejecuta siempre, es decir, no depende de que una variable tenga un valor concreto. Un ejemplo de este tipo de script es en Pueblo Paleta, cuando la mujer aparece mirando el cartel, ya que si miramos en Advance Map, esa no es su posición original. Es importante saber que para mover los minis, debemos usar el comando *movesprite2*, ya que el *movesprite* normal no lo ejecuta.

- **04:** Es exactamente igual que el de tipo 02. Depende de que una variable tenga un valor concreto. Pero en este caso, el script se ejecutará antes de que se cargue la pantalla. Es decir, cuando entras en un mapa, al principio se ve negro, pues el script de tipo 04 se ejecutará antes de que deje de verse negro. Puede ser útil en algunos casos.
- **05:** Es igual que el script de tipo 03, pero si bien este último se ejecutaba cuando se cargaba el mapa (cuando entras en él), el de tipo 05 se ejecutará tanto cuando entres al mapa como cuando salgas de un menú (mochila, pokédex...), inicies la partida guardada en ese mapa o salgas de una batalla.
- **06:** No se conoce el funcionamiento de este tipo de script de nivel.
- **07:** Es igual que el de tipo 03 y 05, pero este no se ejecutará al entrar al mapa, sino sólo cuando salgamos de un menú, iniciemos la partida guardada en el mapa en cuestión, tras una batalla...

Estos son los tipos de scripts de nivel que podemos utilizar y escogeremos uno u otro dependiendo de lo que queramos hacer. Los más importantes son los de tipo 02, ya que son como los scripts de gatillo, que permiten hacer eventos que formen parte de la historia (aparezca el rival para desafiarte, venga alguien a darte algo...). Pero también son los más complejos, ya que en los demás tipos de scripts (01, 03, 05 y 07) bastará con poner el offset del propio script. Por eso los explicaremos detenidamente. Pero antes voy a poner algunos ejemplos para que no queden cabos sueltos.

Imaginemos que en un mapa tenemos un minisprite con el número 2 en las coordenadas (x = 3; y = A), que además el comportamiento que le hemos puesto es que esté mirando alrededor. Pero queremos que cuando se active la flag 0x200, ese minisprite cambie a la posición (x = 5; y = F) y, en vez de mirar alrededor, mire hacia arriba. Utilizaremos un script de nivel de tipo 03 que será tan simple como esto:

```
#org @inicio
Checkflag 0x200
If 0x1 goto @mini
End
```

```
#org @mini
Movesprite2 0x2 0x5 0xF
Spritebehave 0x2 0x7
End
```

En este caso, se ejecutará al entrar al mapa, ya que es un script de tipo 03. Pero si queremos que se ejecute tanto al entrar como al salir de un menú, tras una batalla... Pondremos el de tipo 05. Y si queremos que sólo se ejecute tras salir de un menú, acabar una batalla... Usaremos el de tipo 07. Como he dicho anteriormente, dependerá de lo que necesitemos hacer.

Ahora pondremos un ejemplo con un script de tipo 01 (*setmaptile*). Por ejemplo, en Pueblo



Paleta tenemos un cartel de madera en las coordenadas (x = 5; y = E) y queremos cambiarlo a un cartel normal cuando la flag 0x200 esté activada. Comprobamos en Advance Map cuál es el número del bloque que tiene el cartel normal, en este caso el 2, y creamos nuestro script de nivel de tipo 01 con el siguiente script asignado:

```
#org @inicio
Checkflag 0x200
If 0x1 goto @cartel
End

#org @mini
Setmaptile 0x5 0xE 0x2 0x1
Special 0x8E
End
```

Es tan simple como eso. Claro que siempre adaptado a lo que vosotros queráis hacer. Los scripts de nivel de tipo 01 se utilizan mucho, por ejemplo, para crear un sistema día/noche introduciendo el típico sistema de que las ventanas de las casas se iluminen por las noches.

Ahora vamos con los **scripts de nivel de tipo 02**. Recordemos de nuevo que este script se puede ejecutar en dos casos distintos:

1. Si al entrar al mapa, la variable de la que depende tenga un valor concreto.
2. Si, estando ya en el mapa, se le da a la variable de la que dependa un valor concreto.

Ahora es cuando agradeceremos saber desactivar scripts de gatillo mediante variables y no mediante la opción chapucera de una condición que nos lleve a un pointer con un *end*. Básicamente porque en esta ocasión, si termina el script de nivel pero la variable sigue teniendo el valor que debería tener para ejecutarse el script, este se volverá a ejecutar una y otra vez, es decir, infinitamente. Por eso es necesario, en la mayoría de los casos, darle otro valor a la variable en el propio script. Pero esto no os debería sonar a chino, porque ya deberíais saberlo de los scripts de gatillo.

Bien, como lo mejor es que cojáis un rom limpio para practicar, podéis crear un script sencillo para probar. En el mapa que queráis. Creáis, claro, un script de tipo 02. Ahora escogéis una variable que queráis (y que esté libre). En este caso usaremos la 0x7FFF. Luego en *flag* ponemos 7FFF y en *value* ponemos 0000. Eso quiere decir que el script se ejecutará siempre que el valor de la variable 0x7FFF sea 0x0.

Lo único que queda es asignarle un script, en este caso, en *Script offset 2*. Para probar, pondremos un script normal, con un simple *msgbox*, por ejemplo:

```
#org @inicio
Msgbox @1 0x6
End
```

```
#org @1  
= Probando...
```

Una vez que tenemos configurados los scripts de nivel, **hay que darle a *Save map scripts***. Si vamos al juego y nos dirigimos al mapa de nuestro script de nivel, podrán pasar dos cosas:

1. El script se ejecuta pero se repite continuamente.
2. El script se ejecuta pero después salen letras raras.

En el primer caso, es obvio, ¿No dijimos que había que darle otro valor a la variable para que no se repitiese? Después de que se ejecute el script, la variable 0x7FFF seguirá teniendo el valor 0x0, por lo que el script se volverá a ejecutar. Entonces hay que darle otro valor distinto a la variable, por ejemplo, con un *setvar*. Nuestro script realmente quedaría así:

```
#org @inicio  
Msgbox @1 0x6  
Setvar 0x7FFF 0x1  
End
```

```
#org @1  
= Probando...
```

El script le da el valor 0x1 a la variable, por lo que ya no se volverá a repetir más.

Si es el segundo caso, **se trata de un error muy usual**. Para arreglarlo, vamos al cabezal, seleccionamos la vista profesional del cabezal (se puede hacer de manera rápida pulsando *ctrl+H*) y allí copiamos el offset que hay en *Map script offset*. Ese es el offset que contiene la estructura de los scripts de nivel del mapa. Ahora abrimos XSE y cargamos nuestro rom. Introducimos el offset y descompilamos marcando previamente la opción de *script de nivel* (al lado del botón de descompilar). Ahora nos saldrá la estructura. Lo que tenemos que hacer ahí es buscar algún *#raw word 0xFFFF*, si vemos algo así, lo cambiaremos por *#raw word 0x0*. Eso solucionará nuestro problema y hará que el script se ejecute correctamente.

**Respecto al script de tipo 04**, se configurar exactamente igual, por lo que se puede hacer siguiendo los mismos pasos que para el de tipo 02.

## **•Estructura de los scripts de nivel**

Esto no es necesario saberlo, pero siempre viene bien. Por ejemplo, esto nos permitirá poner más de un script de tipo 02 en un mismo mapa, ya que Advance Map no es capaz de hacerlo de forma automática. Lo que hay que saber es que el script de nivel tiene una estructura principal que guarda el tipo de script y el offset del script. Adicionalmente, la estructura terminará siempre con un *#raw 0x0*. Por ejemplo, imaginemos que queremos poner un script

de tipo 03 con el offset 0x1A456B y otro de tipo 07 con el offset 0x1A4630. Haremos lo siguiente:

```
#dynamic 0x800000
```

```
#org @estructura
#raw 0x3    `Script de tipo 03
#raw pointer 0x1A456B    `Pointer del script de tipo 03
#raw 0x7    `Script de tipo 07
#raw pointer 0x1A4630    `Pointer del script de tipo 07
#raw 0x0
```

Simplemente se pone el tipo del script y debajo su offset. Es bastante sencillo. Pero esto es sólo con los scripts de tipo 01, 03, 05 y 07; los scripts de tipo 02 y 04 no apuntarán directamente al offset, sino a una subestructura propia donde pondremos la variable de la que dependen y el valor que debe tener. La subestructura terminará con un *#raw word 0x0*. Cuidado, con un *#raw word 0x0*, no con un *#raw 0x0* como en es el caso de la estructura principal. Vamos a añadir a la estructura un script de tipo 02 con el offset 0x1A50B0:

```
#dynamic 0x800000
```

```
#org @estructura
#raw 0x3    `Script de tipo 03
#raw pointer 0x1A456B    `Pointer del script de tipo 03
#raw 0x7    `Script de tipo 07
#raw pointer 0x1A4630    `Pointer del script de tipo 07
#raw 0x2    `Pointer del script de tipo 02
#raw pointer @subestructura    `Pointer a la subestructura tipo 02
#raw 0x0
```

```
#org @subestructura
#raw word 0x7FFF    `Variable de la que depende el script
#raw word 0x0    `Valor que debe tener la variable
#raw pointer 0x1A50B0    `Pointer del script de tipo 02
#raw word 0x0
```

Ahora decidimos añadir un script más de tipo 02. En este caso tiene el offset 0x1A5475 y depende de que la variable 0x7FFF tenga el valor 0x2. Lo añadiremos a la subestructura, ojo, a la subestructura, no a la estructura principal:

```
#dynamic 0x800000
```

```
#org @estructura
#raw 0x3    `Script de tipo 03
#raw pointer 0x1A456B    `Pointer del script de tipo 03
#raw 0x7    `Script de tipo 07
```

```
#raw pointer 0x1A4630      'Pointer del script de tipo 07
#raw 0x2      'Pointer del script de tipo 02
#raw pointer @subestructura      'Pointer a la subestructura tipo 02
#raw 0x0
```

```
#org @subestructura
#raw word 0x7FFF      'Variable de la que depende el script
#raw word 0x0      'Valor que debe tener la variable
#raw pointer 0x1A50B0      'Pointer del script de tipo 02
#raw word 0x7FFF      'Variable de la que depende el segundo script
#raw word 0x2      'Valor que debe tener la variable del segundo script
#raw pointer 0x1A5475      'Pointer del Segundo script de tipo 02
#raw word 0x0
```

Es importante saber que si queremos poner un script de tipo 04, también tendrá una subestructura, ya que depende de una variable, pero no es la misma que la de tipo 02, cada uno tiene la suya propia. Para entenderlo mejor, vamos a poner un script de tipo 04 con el offset 0x1A8532 que dependerá de que la variable 0x7FFE tenga el valor 0x1:

```
#dynamic 0x800000
```

```
#org @estructura
#raw 0x3      'Script de tipo 03
#raw pointer 0x1A456B      'Pointer del script de tipo 03
#raw 0x7      'Script de tipo 07
#raw pointer 0x1A4630      'Pointer del script de tipo 07
#raw 0x2      'Pointer del script de tipo 02
#raw pointer @subestructura      'Pointer a la subestructura tipo 02
#raw 0x4      'Pointer del script de tipo 04
#raw pointer @subestructura2      'Pointer a la subestructura tipo 04
#raw 0x0
```

```
#org @subestructura
#raw word 0x7FFF      'Variable de la que depende el script
#raw word 0x0      'Valor que debe tener la variable
#raw pointer 0x1A50B0      'Pointer del script de tipo 02
#raw word 0x7FFF      'Variable de la que depende el segundo script
#raw word 0x2      'Valor que debe tener la variable del segundo script
#raw pointer 0x1A5475      'Pointer del Segundo script de tipo 02
#raw word 0x0
```

```
#org @subestructura2
#raw word 0x7FFE      'Variable de la que depende el script
#raw word 0x1      'Valor que debe tener la variable
#raw pointer 0x1A8532      'Pointer del script de tipo 04
#raw word 0x0
```

Y esa sería la estructura complete de los scripts de nivel de nuestro mapa. Recordad que cada mapa tiene su propia estructura y, por tanto, sus propios scripts de nivel.

Lo único que hay que saber diferenciar es que los scripts de tipo 01, 03, 05 y 07 no dependen de ninguna variable y se ejecutan siempre, por lo que en la propia estructura principal llevarán el pointer al propio script. Por otra parte, los de tipo 02 y 04 necesitan la variable y el valor, por ello necesitan una subestructura.

*“Scripts de gatillo y de nivel”*

Manual redactado por **Javi4315**.

Queda estrictamente prohibida su distribución.