Scroll y transiciones de escenario en NintendoDS

Curso 2014/2015

Apellidos, nombre	Luis Fábregues de los Santos (luifbde@inf.upv.es)
Titulación	Grado de Ingeniería informática
Fecha	01/04/15

UNIVERSIDAD POLITECNICA DE VALENCIA

Índice


UNIVERSIDAD
POLITECNICA
DE VALENCIA

UNIVERSIDAD POLITECNICA DE VALENCIA

Índice de ilustraciones

1 Introducción

A mediados de 1983 la empresa *Atari*, la que en aquella época era líder del mercado de videoconsolas, entró en crisis. Esto se debió en mayor parte a varios años de publicar enormes cantidades de juegos de baja calidad, haciendo que los consumidores perdieran fe y confianza en este sector.

No fue hasta un par de años después que *Nintendo* lanzó lo que muchos consideran la mejor consola de la época, la *Nintendo Entertainment System (NES) o Famicom* en japón. Lo que sí que es un hecho es que en 1987 esta consola se convirtió en un éxito de masas en Estados Unidos. Quizás el panorama actual de los videojuegos no sería igual sin la aparición de aquella máquina.

Fuera de su influencia, esta consola fue una maravillosa vaina de nacimiento para los juegos conocidos como *sidescroller*. Títulos como *Super Mario Bros, Megaman, Metroid, Castlevania, Defender, Lifeforce, Contra, Kid Icarus, Blaster Master, Ninja Gaiden* y la lista continúa. La cantidad de este tipo de videojuego en las consolas actuales no se queda siquiera cerca de los que se pueden encontrar en la *NES*, debido en mayor parte a que se prefieren juegos que aprovechen las capacidades de renderizado en 3D de una forma más impresionante a la vista.

Los *sidescroller* tienen unas cuantas cosas en común. Son juegos que aplanan una de las dimensiones para no tener que representar entornos tridimensionales. Suelen emplear vistas laterales para ver a los personajes, de perfil, avanzar a izquierda o derecha aunque también existe una gran cantidad de juegos en los que se ve a vista de pájaro y los personajes se mueven por toda la pantalla sin restricciones.

Para que el juego de la sensación de transición, se utilizan diferentes fondos o escenarios para caracterizar los sitios a los que el personaje principal puede acceder. Este trabajo se centrará en identificar cómo se realizan estas transiciones de escenario e intentar crear una réplica de demostración para la *NintendoDS*.

Ilustración 1: Metroid, juego de NES

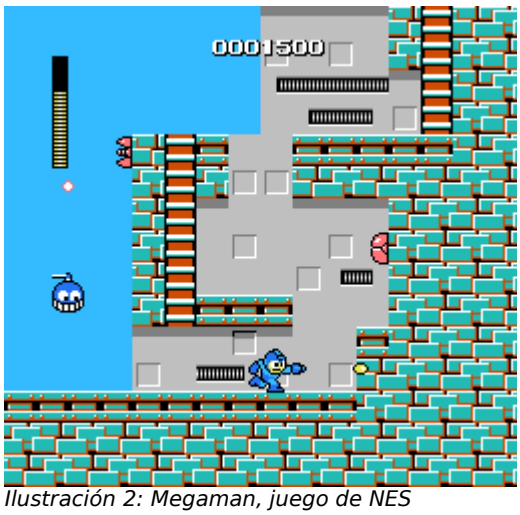

Ilustración 2: Megaman, juego de NES

2 Objetivos

El primer objetivo será realizar un análisis de las técnicas de cambio de escenario utilizadas en diversos juegos de *scroll* horizontal y vertical para posteriormente investigar cómo se podrían implementar dichas funciones en *NintendoDS.* Se hablará de juegos que marcaron el comienzo de una nueva era para estas aplicaciones lúdicas entre las que se incluyen muchos títulos famosos como *MegaMan, Metroid, Castlevania* y *Legend of Zelda.*

Como segundo punto se propone indagar en las funciones que ofrece *DevKitPRO* para realizar este tipo de implementaciones. Con especial énfasis en los fondos, la investigación se centrará en cargar *tilesets* y paletas para luego construir un escenario con dichos recursos y así terminar identificando la forma de realizar transiciones de manera robusta.

La tercera parte de este proyecto consistirá en implementar dichos fondos y técnicas de transición. Se analizará su viabilidad y se expondrá de forma clara como dicho experimento puede ser replicado y usado como punto de apoyo en futuros proyectos de esta misma índole. Para ello se implementará un ejemplo simple tanto como de *scroll* horizontal como de *scroll* vertical.

3 Desarrollo

El desarrollo se centrará en los tres apartados mencionados en el objetivo: investigación sobre técnicas de cambio de escenario, indagar sobre las herramientas para implementar dichas técnicas y finalmente implementarlas.

3.1 Técnicas de cambio de escenarios

Como ya se comentó en la introducción, los más excelsos ejemplos de *scroll* fueron desarrollados para la plataforma *NES* y las técnicas que en ellos se ven han sido replicadas durante muchos años y diversas en generaciones de plataformas posteriores. En este apartado se hará un breve paseo por los títulos más famosos de dicha consola, las técnicas que se pueden apreciar en ellos y las influencias que ejercen en videojuegos más recientes.

Se comienza analizando la más simple de las técnicas que se pueden encontrar en un título de *NES* que, irónicamente, acabó siendo una de las mayores franquicias que Nintendo posee actualmente. Este juego es sin duda, *The Legend of Zelda.* A pesar de que este título no entra en la categoría de *sidescroller* como tal, sí que se puede apreciar una de las técnicas más simples de transición de escenario. Cuando el personaje, *Link*, entra en contacto con los bordes de la pantalla y esa pantalla tiene salida, se observa que: los enemigos desaparecen, se pierde el control del personaje, el siguiente escenario comienza a aparecer por el lado al que se quería acceder y el escenario en el que estaba el personaje comienza a desaparecer por el lado contrario. Esta transición dura hasta que el escenario antiguo ha sido totalmente substituido por el nuevo y durante este tiempo no se podrá controlar al personaje. Cabe comentar que todas las técnicas de transición vistas tienen el objetivo de representar visualmente que el personaje accede a otra zona del juego, por tanto la transición se suele hacer de tal forma que parezca que el personaje ha cruzado ese lugar caminando. Por motivos de comodidad se denominará está técnica Tipo 1 en partes posteriores de esta memoria.

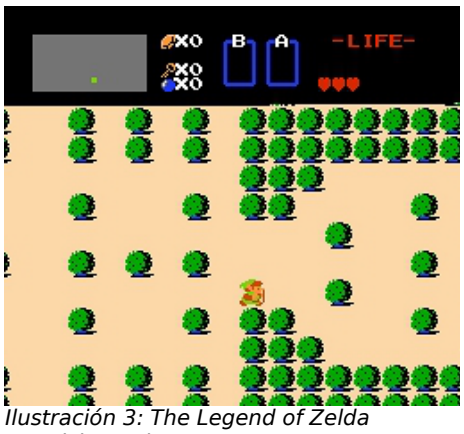

Ilustración 3: The Legend of Zelda transicionando.

Pese a que la técnica vista anteriormente sea suficiente para desarrollar juegos tan maravillosos como el que se acaba de exponer, los programadores raras veces resisten la tentación de rizar el rizo y llegar a juegos con capacidades gráficas más bellas a la vista. Se presenta *MegaMan* un título que introdujo grandes variaciones a los juegos de tipo *scroll* vistos en esta misma plataforma. El juego es uno de los *sidescrollers* más tradicionales y famosos de esta consola. La técnica de transición es ciertamente parecida a la anterior, pero la pequeña modificación que trae hace las transiciones mucho más suaves y creíbles. En este juego la cámara (o punto de vista del usuario) parece centrarse en el personaje principal (en este caso *MegaMan* o *RockMan* en su versión japonesa) siguiéndolo allá a donde va y cargando el escenario a su alrededor según anda. Sin embargo, en la mayoría de juegos que usan esta técnica, hacen uso de la técnica Tipo 1 para cambiar entre bloques de escenarios. A esta técnica se denominará a partir de ahora Tipo 2.

Quizás sea importante, antes de proseguir, hacer un pequeño hincapié en el que quizás sea el juego más conocido de la historia y analizar su técnica de *scroll*. Este es sin duda *Super Mario Bros.* en *NES*. En todos los escenarios se ve una simplificación de la técnica Tipo 2 que impide que el personaje se mueva hacia atrás. Quizás esto parezca una restricción a la técnica anterior, pero como se verá más adelante la implementación resulta mucho más fácil en este tipo de juegos.

Es momento de cambiar de tercio a un tipo de juego diferente en apariencia pero no tan distinto a un nivel de implementación de los que se han encontrado hasta ahora. *LifeForce* donde el jugador controla una nave espacial en un entorno de movimiento sin restricciones en *scroll* horizontal. Se observa que esta técnica es prácticamente igual la que se ha analizado en *Super Mario Bros*, la nave se desplaza siempre hacia la derecha y sin poder retroceder. La mayor diferencia, a parte del control del personaje principal, es que el escenario está en constante movimiento.

Ahora se verá un pequeño cambio de paradigma, concretamente al scroll vertical. Una vez más, el clásico de *NES LifeForce* tiene esta técnica de transición. La diferencia visual es bastante significativa, sin embargo se puede comprobar que estos juegos suelen utilizar técnicas muy similares a la de Tipo 2. Un asunto a considerar es que muchos de estos títulos utilizan fondos cíclicos o aleatorizados, aunque este en concreto no lo haga. Las diferencias de codificación son pocas a nivel conceptual, pero existen varias discrepancias que se discutirán en el apartado correspondiente a la implementación. Se codificará la técnica de *scroll* vertical como Tipo 3.

Ilustración 4: LifeForce en scroll horizontal

3.2 Prestaciones de *DevKitPRO* para escenarios

Como se vio en escasa profundidad en el tutorial *How To Make A Bouncing Ball Game*, *DevKitPRO* permite la carga de imágenes en formato *png* que posteriormente son procesadas por la herramienta *Grit* para que sean utilizables en la compilación del proyecto añadiendo una serie de cabeceras que determinan la paleta de color, dimensiones, etc. Una vez se dispone de dichas cabeceras, se puede hacer referencias a ellas en el código para que se puedan dividir en *tiles.* Una vez se dispone de *tiles* es trabajo del programador colocarlos en un *array* que representa el mapa del fondo. La colocación de dichos *tiles* en dicho *array* conformará la apariencia del fondo que aparecerá pantalla. Existe todo un elenco de detalles necesarios para llegar a mostrar un fondo por la pantalla de *NintendoDS* pero estos detalles se pueden encontrar en el mencionado tutorial.

Los apartados más interesantes que demuestra este ejemplo son: la carga de fondos, el dibujo mediante *sprites* en el *array* de fondo, el paradigma de la repetición del fondos y los registros de *scroll*.

La carga de fondos, pese a parecer mecánica dado el material de apoyo, tiene ciertas restricciones. En un primer momento se intentó cargar una imagen que ocupara toda la pantalla de la *NintendoDS*. Esta primera idea trataba de ahorrar los costes de tener que crear (o encontrar) un *tileset* y construir escenarios mediante él. A pesar de que la imagen fue cargada con éxito y el fondo se veía perfectamente, surgió un problema. Al cambiar el valor del registro de *scroll* vertical aparecía basura en la imagen en la parte que la pantalla verticalmente no abarca. Este pequeño problema hubiera podido ser “barrido bajo al alfombra”, pero se optó por una implementación usando *tilesets* para evitar futuros problemas y para mantenerse fiel a las técnicas usadas en el pasado.

El dibujo mediante *sprites* no va mucho más allá de lo que se vio en *Bouncing Ball,* para poder usarlo correctamente en este proyecto solo se fue necesario la creación de un *tileset.* Se utilizó una captura de pantalla de *Megaman 1* y se construyó mediante *Gimp.*

Uno de los factores más determinantes para comprender las transiciones entre escenarios en la *NintendoDS* es conocer que los fondos son replicados automáticamente por la consola. Esto puede suponer un gran problema a la hora de hacer transiciones, ya que cuando se desplaza el fondo mediante cualquiera de los registros de *scroll* se verá aparecer el mismo fondo por el otro lado. Esto pasa porque se estará trayendo un réplica del mismo si no se hacen diferentes variaciones. Como se explica en el mencionado tutorial se dispone de un registro de tamaño. Este registro permitirá que se pueda dibujar un fondo sin que este se tenga que ver obligatoriamente en la pantalla, haciendo posible la tarea de cambiar de escenario de forma fluida.

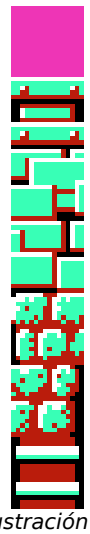

Ilustración 5: Tileset

Por último, los registros de *scroll* de los que dispone *LibNDS* son extremadamente importantes. Sin ellos la transición se debería hacer a *tiles* dando una impresión muy aparatosa y poco fluida (o realizar un pintado de escenario a píxeles). Como se verá en el siguiente apartado, estos registros dan un juego indispensable para alcanzar los objetivos.

3.3 Implementación

Se comienza con la aplicación desarrollada en el tutorial de *Bouncing Ball.* Tras unas pequeñas modificaciones en el código se consiguió que la pelota dejara de rebotar en el suelo y se habilitó un botón de salto. Para ello se aprovechó la implementación que ya se había hecho, eliminando ciertas funciones y añadiendo en el archivo *main* una función que, al pulsar arriba (nuestro botón de salto), añadiera la velocidad adecuada a la pelota. El siguiente paso fue construir el fondo mediante bucles y operaciones de *bitshift*. A pesar de que la implementación podía haber sido mucho más simple utilizando otro tipo de algoritmos, se ha optado por este método por ser más fácil a la hora de probar modificaciones en el mismo. Todo esto está comentado en el código, por tanto no se hablará más de esta parte.

También cabe considerar que la carga y el manejo de *sprites* quedan fuera de este trabajo. Se hablará de la lógica necesaria para mover estos recursos pero se hará uso del *sprite* de la pelota de *Bouncing Ball.*

3.3.1 Tipo 1

El objetivo de este apartado será recrear un pequeño conjunto de tres escenarios a los que el personaje podrá acceder tocando los bordes de la pantalla con el tipo de transición que ya se ha descrito.

Antes que nada, se debe indicar a *LibNDS* que se va a utilizar un fondo que ocupará dos bloques, ambos horizontales. Para ello se utilizará la instrucción *REG_BG0CNT = 1 << 14 | 1 << 8;* que, aparte de indicar a la consola que se desea un fondo de doble dimensión horizontal, respeta el *screen base block* definido en el tutorial. Este cambio hará que el *array* de mapa de fondo (*bg0map*) considere índices hasta el 2047 y represente los fondos de manera acorde. También cabe destacar que los *tiles* se representarán como si fueran dos *arrays* por separado, es decir, que los *tiles* del fondo que se ve en la pantalla con *REG_BG0HOFS==0* son los que ocupan las posiciones de la 0 a la 1023 y los que aparecen en el otro fondo ocupan las posiciones de la 1024 a la 2047.

Como también se discutió en el tutorial del *Bouncing Ball* todas las estructuras de datos que se utilicen deben de ser inicializadas previamente. En este caso se recorre el *array bg0map* desde la posición 0 hasta la 2048 (sin incluir) asignando sus valores a 0.

Para ser capaces de mantener un sistema de *scroll* consistente, se tiene que llevar una serie de variables que determinen qué fondo se debe cargar, dónde y en qué punto de la transición se encuentra el programa. Se definirá la variable *scrolling* inicializada a cero que servirá de baliza para determinar si se está realizando una transición y de qué tipo es. Esa variable tendrá el uso de prohibir la entrada del usuario (como se vio en el apartado 4.1 este tipo de transición evita el movimiento del personaje). También se definirá la variable *horizontal* que determinará en qué punto de la transición entre escenarios está el programa. Una variable llamada *actualStage* mantendrá constancia sobre el escenario en el que se encuentra el personaje principal.

Seguidamente se crea una serie de métodos que carguen un fondo en una pantalla concreta, ya que será de especial utilidad que el programa sea capaz de cargar con una llamada simple el fondo del escenario al que se accederá haciendo esta técnica de *scroll*. Estos métodos tiene la forma de *drawXinY*.

En el apartado de procesar la lógica se incluye una función que compruebe si el objeto *character* está tocando cualquiera de los bordes horizontales de la pantalla, porque de esta forma se activará el *scrolling.* Estas instrucciones determinarán si se debe aplicar una transición o no (por ejemplo, en el primer escenario no se puede ir a la izquierda, por tanto no se activará ninguna transición) y si se está intentando acceder al escenario izquierdo o al derecho.

En caso de estar accediendo a un escenario en la derecha de la posición actual se incrementa la variable *horizontal* que se asignará al registro *REG_BG0HOFS*. Que esta variable se incremente positivamente se traducirá en un desplazamiento hacía la izquierda del fondo, dando la impresión de que se está avanzando a derechas. Este incremento se detendrá si el escenario nuevo está completamente en la pantalla y esto se comprueba cuando la variable *horizontal* supera el valor 256. Se puede variar a la velocidad a la que el escenario se desplaza si se varía el incremento de esta variable. Cuando el *scroll* haya terminado, incrementará *actualStage* para expresar que se ha alcanzado el siguiente escenario y asignará el valor las variables *scrolling* y *horizontal* a cero, ya que estas solo se usan para transicionar.

Si se avanza a un escenario a la izquierda del personaje principal, primero se asignará *horizontal* a 256, por tanto nuestro punto de vista se quedará en el segundo fondo. Para que la información en la pantalla se mantenga consistente se debe cargar el escenario actual en el segundo fondo, pero eso se explicará más adelante. El resto de la función es bastante similar al *scroll* a derechas, solo que *horizontal* decrementará y la transición terminará cuando *horizontal* sea cero. Al final del código correspondiente se pondrá a cero el valor de *scrolling* y se decrementará *actualStage*.

Hasta ahora se ha hablado del movimiento de fondos, pero para mantener la sensación de transición también se debe mover el personaje, ya que en caso de dejarlo en el mismo sitio no daría el efecto buscado. Esto puede parecer poco intuitivo, pero en la práctica el personaje se tiene que mover junto a la pantalla para dar la sensación de desplazamiento global. Para realizar este movimiento sobre el personaje se ha creado una función nueva dentro de *character.c* (antiguamente *ball.c*) que mueve al personaje sin tener en cuenta las velocidades ni la gravedad.

Finalmente, en *updateGraphics* gestionará los mapas que deben de ser cargados según la variable *scrolling.* Si se accede al escenario de la derecha se cargará el escenario actual en el fondo 1 y el escenario siguiente en el fondo 2. Si accede al escenario de la izquierda cargará en el fondo 1 el escenario al que se accede y en el fondo 2 el escenario en el que se encuentra el personaje. Como se ha descrito anteriormente la lógica se encargará de situar la cámara en el fondo que toca.

En resumen, el funcionamiento del algoritmo es el siguiente. Cuando se quiere acceder a un escenario que está a la derecha, este escenario se carga en las posiciones del array a partir de la 1024. Una vez echo esto, se podrá utilizar el registro *REG_BG0HOFS* para que el fondo antiguo desaparezca por la izquierda y el fondo nuevo aparezca desde la derecha. Este registro se incrementará hasta que haya alcanzado el valor de 256, indicando así que el nuevo fondo aparece en la pantalla completamente. En caso de desplazarse hacia la izquierda se cargará el fondo antiguo en el array a partir de la posición 1024 y el nuevo en las posiciones anteriores. Se actualizará *REG_BG0HOFS* al valor 256 y se decrementará constantemente hasta completar la transición.

3.3.2 Tipo 2

Utilizando la misma configuración de fondos que en el apartado anterior, se hablará de como desplazar al personaje por los tres escenarios de forma semejante a la que se vio en la descripción de este tipo de transición en el apartado 4.1. Este desplazamiento será más gradual y más elegante. Se basará en el código desarrollado en el apartado anterior.

Las variaciones de este tipo de *scrolling* tienen más relación con las flechas direccionales que con el estado puro de la máquina. Si el usuario pulsa el botón de la derecha y el personaje está muy alejado del centro de la pantalla se debería mover, pero el fondo no. Por el contrario, si el personaje está en el centro de la pantalla lo que se debería mover sería el fondo para dar la sensación de desplazamiento. Para implementar estas mejoras se comentarán primero detalles relativos a las modificaciones fuera del *input* del usuario para luego dar una definición más contundente de lo que pasa cuando el usuario introduce datos.

Una nueva modificación será que se considera que el personaje está en el escenario 1 mientras no haya llegado a *horizontal == 256.* Así, por tanto, solo habrá dos estados de *actualStage,* cero y uno, correspondiendo el primero al fondo cero y el fondo uno y el segundo al fondo uno y al fondo dos.

Horizontal sigue cumpliendo la misma función que en el ejemplo anterior y en *processLogic* ahora se detectará si horizontal ha superado 256. Si esto pasa y *actualStage==0* significará que se está accediendo al escenario 2, en caso contrario querrá decir que el programa está al máximo del escenario 2 y se debería prohibir que el fondo se desplace (además de dejar que el personaje se mueva libremente). Algo parecido pasa cuando *horizontal* es menor que 0. En ese caso si el personaje está en el escenario 1 se le dará libre movimiento y se bloqueará el movimiento de los fondos. Si el escenario es el 2, se indicará que la ejecución se encuentra en el estado 1.

En *updateGraphics,* ya que se disponen de solo dos estados, solo será necesario cargar el escenario actual y el siguiente, en caso de que la ejecución no se encuentre en el último escenario. También se actualizará el *REG_BG0HOFS* con *horizontal*.

Cuando la tecla derecha o izquierda es pulsada se comprueba si el personaje tiene libre movimiento o si se debe mover el fondo. Para comprobar esto se debe considerar si no hay más escenario en esa dirección o si aún no se ha alcanzado la mitad de la pantalla para mover el fondo. Si cualquiera de las dos condiciones anteriores se cumplen, el personaje tendrá movimiento libre.

Como se dijo en la introducción, el tipo de *scroll* usado en *Super Mario Bros* es bastante más simple que este, como se habrá podido comprobar. Esto se debe a que no se tiene que tener en cuenta más que la mitad de este problema, el desplazamiento a la derecha. El personaje principal solo tendrá libertad de movimiento cuando el usuario intente ir a la izquierda y la implementación no se tendrá que preocupar se cargar ningún fondo por algún lado que no sea el derecho.

En conclusión, el código se encargará de comprobar si el personaje ha alcanzado al mitad de la pantalla. En caso de hacerlo, cada paso que de el personaje debería mover el fondo (y no el *sprite*) hasta alcanzar la mitad de la última pantalla. Si el personaje se encuentra a la izquierda de la mitad de la primera pantalla o a la derecha de la mitad de la última debería moverse sin desplazar el fondo.

3.3.3 Tipo 3

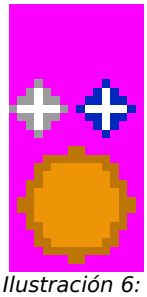

Ilustración 6: Tileset tipo 3

Esta parte del proyecto tuvo un matiz adicional ya que no fue posible encontrar *sprites* ni *tilesets* de juegos de naves de *scroll* vertical que armonizara con la idea que se tenía a la hora de implementar esta parte, por tanto se optó por diseñarlos con *Gimp.*

La idea de esta parte pudiera haber sido exactamente la misma que en los dos tipos anteriores: activar un fondo de doble tamaño vertical y representar los objetos que se quieren visualizar cuando ocurra la transición. Pese a que hubiera sido efectiva y totalmente viable optar por esta implementación, la práctica con esta consola ha demostrado que hay mejores formas de hacer esto.

El registro de *scroll* que se comentó en anteriores apartados ahora será reemplazado por su equivalente vertical, *REG_BG0VOFS.* Como bien es sabido, la pantalla superior de la *NintendoDS* no es cuadrada, por tanto existe una parte del mapa de fondo que no se puede ver en la pantalla. Se constituye de 256 entradas, por tanto se dispone de una parte del fondo con altura de 8 *tiles* que no se representa en la pantalla a no ser que se modifique el valor del registro de *scroll* vertical. Esta información será relevante para hacer una implementación algo más complicada pero que ahorra memoria a la consola.

Se realiza una pequeña demo con una nave que sea capaz de volar dentro de la pantalla y un fondo cargado aleatoriamente. El objetivo será dar la sensación al usuario de que se está avanzando por el espacio. Para esta versión se reconsiderarán ciertas variables: *horizontal* se cambiará por *vertical* por razones de consistencia y se prescindirá de *actualStage* al carecer de escenarios predefinidos. También se harán un par de cambios menores en el código de *character.c* para no considerar los efectos de la gravedad ni el contacto con el suelo.

No se comentará en esta memoria nada referente a la creación del fondo aleatorio ni a las variaciones al archivo *character.c* para conseguir los efectos visuales básicos. La forma de hacer esto es fácilmente deducible de *How to make a Bouncing Ball* y del código creado.

Para comenzar se fijará la variable *vertical* a 64, ya que este valor mantiene fuera de la pantalla la parte superior del mapa de fondo. Por otro lado, disminuir esta variable hará que el fondo baje, dando la sensación de movimiento. En este caso la variable se asignará a *REG_BG0VOFS* en el método *updateGraphics.*

Dentro de *processLogic* se puede encontrar el grueso de esta implementación, a pesar de que al ser un fondo aleatorio la dificultad es pequeña. En esta función simplemente se disminuirá la variable *vertical (*ya que el fondo se moverá siempre) y posteriormente se comprobará si esta variable ha alcanzado el valor cero. En este caso se redibujará el fondo y se asignará la variable *vertical* a su valor inicial, 64. Esto dará la sensación de avance ininterrumpido de forma eficiente.

La función de redibujo del fondo será donde se encuentre la verdadera complejidad de esta implementación. Para mover la parte que se debe seguir representando en la pantalla se usará un bucle desde la posición 767 hasta la cero del *array* del mapa de fondo. Este bucle moverá los *tiles* hasta las posiciones mayores (que son las que se representan en el inferior de la pantalla) para mantener la representación consistente. Posteriormente se usará un bucle para llenar las posiciones desde cero hasta 256 con nuevos *tiles* aleatorios que aparecerán en la pantalla cuando el registro vertical decremente.

Si se desea hacer un fondo predeterminado y no aleatorio se debería usar una representación como un archivo de números o letras. El algoritmo debería ir cargando fragmentos de ese archivo para dar la sensación de recorrido. Cargar objetos de gran tamaño es trivial con esta implementación.

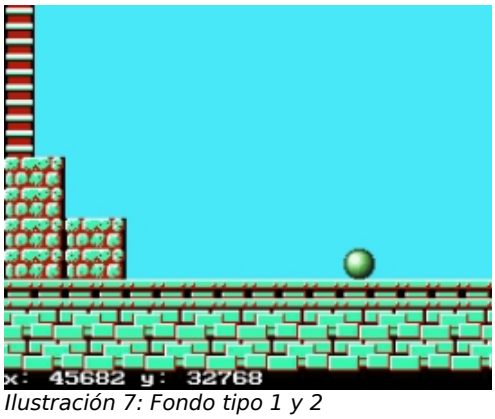

Ilustración 7: Fondo tipo 1 y 2

Ilustración 8: Fondo tipo 3

4 Conclusión

Se ha logrado los objetivos propuestos. Las técnicas de *scroll* más comunes (hasta donde llega el conocimiento del autor) se han resumido y analizado en este trabajo. Se ha observado algunas de las características del entorno de desarrollo que permiten dicho trabajo y se han hecho observaciones sobre como utilizarlas. Finalmente se ha hecho un código de muestra con cada una de estas técnicas funcionando y explicado de forma que se pueda replicar.

El proyecto ha sentado las bases de las formas de *scroll* más comunes con una forma sencilla de implementación y cuyas bases servirán, con suerte, a proyectos futuros de esta misma temática. También da varios ejemplos de carga de fondos y movimiento de *sprites,* por tanto también se considera un trabajo de valía ejemplificadora para el desarrollo en esta consola.

Las investigaciones sobre este tipo de juego no están en su completud. El desarrollo de mejoras debería centrarse en la implementación de plataformas, interacción con el escenario y la creación de enemigos. El trabajo está lejos de completo, pero se espera haber dado un gran punto de apoyo para futuros trabajos en este ámbito.

5 Bibliografía

[1] Tutorial: How to make a bouncing ball → http://ekid.nintendev.com/bouncy/index.php

[2] Libnds documentation → http://libnds.devkitpro.org/

[3] Materiales usados para crear el *tileset* → http://www.bghq.com/