



# ***Gestión del sonido en la implementación del tutorial “The Bouncing Ball” para NDS***

<b>Apellidos, nombre</b>	<b>Agustí Melchor, Manuel</b> (magusti@disca.upv.es)
<b>Departamento</b>	<b>Dpto. de Ing. De Sistemas y Computadores (DISCA)</b>
<b>Centro</b>	<b>Escola Tècnica Superior d'Enginyeria Informàtica</b> Universitat Politècnica de València

## 1 Resumen de las ideas clave

En este artículo, vamos a recorrer parte de la creación de Mukunda Johnson [1], quien realizó desarrollos *homebrew* (los que utilizan el kit de desarrollo no oficial) muy interesantes para Nintendo DS (NDS). Entre otras cosas, realizó un tutorial<sup>1</sup> sobre el desarrollo de aplicaciones para NDS que permite iniciarse en este complejo mundo. Sobre el trabajo de Johnson, Jaén [2] realizaría un trabajo de actualización y ampliación del tutorial original para seguirlo, al tiempo que se corrigen algunas erratas del texto original, algunos cambios del código debidos a modificaciones de las librerías en que se basa y ampliando algunos apartados con pequeñas “demos” de conceptos.

Ya no es posible consultar el tutorial original en la red, así que el presente trabajo ofrece una versión conjunta de estos dos trabajos citados, actualizándolos a las versiones presentes de las librerías sobre las que se basan. Para facilitar el seguimiento de este trabajo se han dispuesto todos los elementos del proyecto que aquí se comenta en GitHub [3], junto a los documentos originales mencionados. Quiero rendir un homenaje a estos desarrolladores que han contribuido con su código, y con sus explicaciones, a que otros puedan adentrarse en este difícil campo de desarrollo de aplicaciones para videoconsolas, que son plataformas muy populares y diferentes del ordenador de sobremesa.

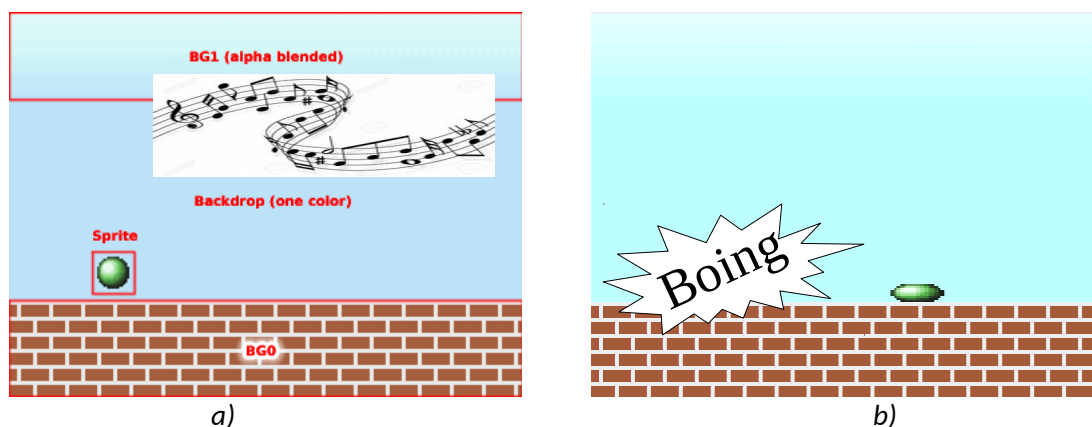


Figura 1: Esquema gráfico de las dos situaciones donde introducir elementos sonoros: (a) música de fondo y (b) un efecto puntual: rebote de la pelota.

Tomaremos como base para este artículo la aplicación para la NDS que explican los trabajos de [1] y [2]. En ella, sobre una escena con dos planos y utilizando la botonera de la NDS, se puede mover una pelota, a la que también afecta el efecto de la “gravedad” y el “rozamiento con el aire” que la hacen avanzar en una trayectoria casi parabólica y que, al tocar el suelo, sufre una deformación y rebota. Nos ocuparemos en este artículo de comentar las acciones necesarias para añadir a esa aplicación una música de fondo (a veces llamada banda sonora o BS) y un sonido puntual (clip de audio, efecto especial o FX); la Figura 1 muestra dos bocetos para ilustrar estas situaciones. Asumiremos que la aplicación ya está desarrollada y nos centraremos en los aspectos concernientes al uso del audio como recurso.

<sup>1</sup> Publicado originalmente en “How to Make a Bouncing Ball Game for Nintendo DS” <<http://ekid.nintendev.com/bouncy/>>, este trabajo ya no está disponible en la red.

## 2 Objetivos

Una vez que el lector haya leído con detenimiento este documento:

- Podrá experimentar con un código que utiliza los conceptos básicos de desarrollo de aplicaciones de videojuego en la consola NDS, centrándose en el uso del sonido para esta plataforma.
- Será capaz de entender el proceso de transformación de los recursos de naturaleza sonora que se incorporan a una aplicación para la plataforma NDS.

No es un objetivo instalar las herramientas que permiten la creación del ejecutable y la carga del mismo, por lo que asumiremos que se dispone del emulador *DeSmuMe* [4]. Si está interesado en los detalles de estos aspectos se puede recurrir a los trabajos de [5] y [6]. Tampoco es un objetivo de este trabajo entrar a ver las características de un computador optimizado para tareas de videojuego. Si el lector tiene curiosidad o necesidad de profundizar en los elementos de la NDS, es recomendable consultar [7].

Antes de ver un ejemplo, hemos de hablar un poco de teoría, en concreto cómo se utiliza el audio en la NDS. Veamos qué consideraciones hay que tener en cuenta para poder generar sonidos a partir de ficheros en esta videoconsola.

## 3 Introducción

En el ámbito de los videojuegos se habla habitualmente de dos conceptos relativos al sonido en las aplicaciones que se desarrollan y que son:

- Sonidos. Que hace referencia a la síntesis de sonidos simples que pueden sonar en múltiples instancias y a la vez. Estos sonidos suelen cargarse completamente en memoria (y descargarse cuando no se requieren).
- Música. Hace referencia al sonido del solo hay una instancia en reproducción y que hay que llevar cuidado ya que su tamaño es importante y, por tanto, sus requerimientos de memoria.

### 3.1 Arquitectura de la NDS: soporte de audio

La NDS ofrece, para la salida de audio, 16 canales *hardware* independientes: esto es, que pueden funcionar con parámetros diferentes de frecuencia de muestreo, tamaño de muestra y número de muestras; y se pueden habilitar de manera separada, ser reproducidos en modo bucle o no, variar su volumen, etc. Todo ello con un sistema basado en 2 canales (estéreo) y balance (*panning*) entre ellos, que funciona a una frecuencia máxima de muestreo 22 Khz. También dispone de un micrófono que proporciona una señalmonofónico, de 8 o 12 bits PCM. El subsistema *hardware* de sonido de la NDS está compuesto por tres elementos. El primero se denomina *Simple Sound Engine* (SSE) que proporciona las operaciones básicas de manipulación de sonido (como volumen o balance) y el acceso al micrófono. Además, dispone de un *Programmable Sound Generator* (PSG), que proporciona la generación de formas de onda simples (o

tonos) y, por último, también dispone de un “generador de ruido blanco” (*noise generator*) para sintetizar sonidos más complejos.

El formato nativo de sonido en la NDS que conoce habitualmente como *raw format*, está basado en muestras codificadas en forma de ondas, sin comprimir, en PCM / ADPCM, de 8 o 16 bits. Así que deja de mano del software la reproducción de formatos más complejos y el acceso a ficheros. Aquí veremos ejemplos de uso de estos segundos y como de la mano de la librería *Maxmod* [1] se pueden reproducir formas de ondas (*sample-playing*), reproducir ficheros estructurados (*mod-playing*) y gestiona el modo *streaming* (buffers).

## 4 Estructura y contenido del proyecto

El proyecto tiene la apariencia visual que se muestra en Figura 2a y se estructura en la forma en que se muestra en la Figura 2b, con un subdirectorio para el código fuente (*source*), otro para los recursos gráficos en formato de mapas de bits (*graficos*) y otro para el audio (*sonidos*). La Figura 2c muestra estos últimos, ampliados para que se vean los detalles y que son los únicos que utilizaremos en el desarrollo de la escena. Además en el directorio del proyecto, hay un fichero *Makefile* que es el encargado de organizar el trabajo a realizar con los ficheros para obtener el ejecutable final. Para ello, en él se configuran los subdirectorios que contiene cada recurso (código fuente, imágenes y sonidos) y las operaciones a realizar con cada herramienta para convertir y combinar todos los recursos en el fichero resultante final.

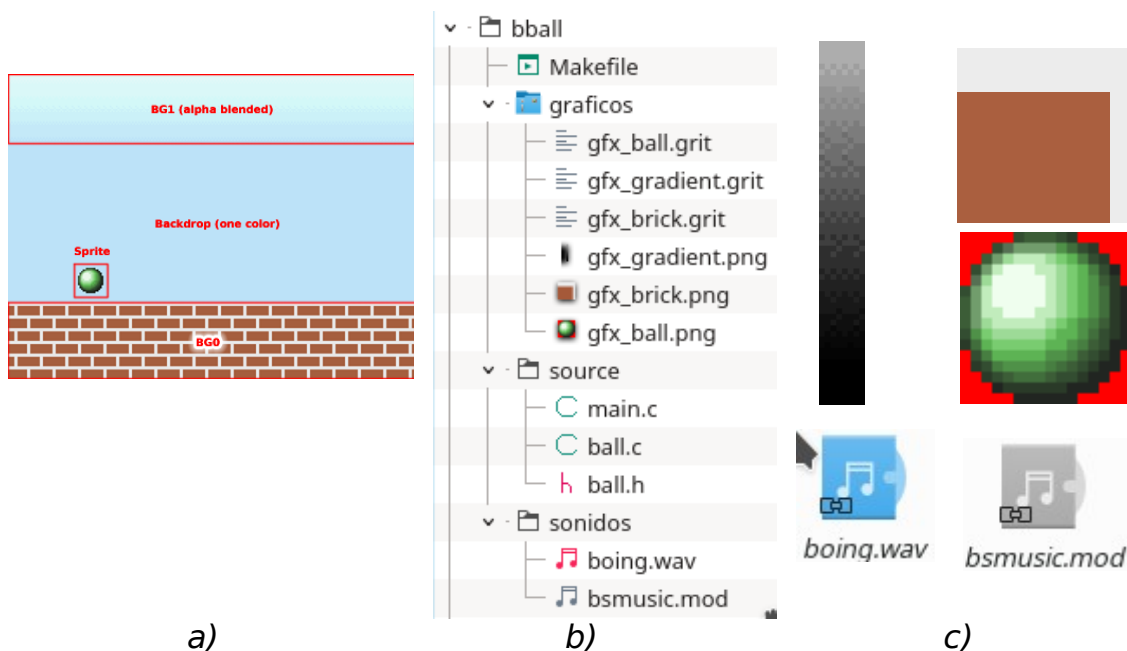


Figura 2: El proyecto: (a) imagen final del tutorial extraída de [2], archivos que lo componen (b) y (c) previsualización de los recursos gráficos y sonoros.

Los recursos sonoros están en el subdirectorio “sonidos” del proyecto, Error: no se encontró el origen de la referenciab y se han obtenido de repositorios libres ([8] y [9]). Sus características son las que muestra la Figura 3. El fichero de sonido original no es posible utilizarlo en la NDS por sus altos requerimientos, especialmente la frecuencia de muestreo de 44 Khz que es mucho mayor que lo que soporta la consola. Por ese motivo, hay que convertirlo a unos parámetros más sencillos. Sin perder calidad para su propósito, con Sox

[8] hemos obtenido una versión apta para ser utilizada y el lector interesado podrá probarlo, si tiene curiosidad.

```
$ file bball/sonidos/*
boing.wav: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 8
bit, mono 10000 Hz
bball/sonidos/bsmusic.mod: 4-channel Protracker module sound data
Title: "elekfunk !"

$ file 140867__juskiddink__boing.wav
140867__juskiddink__boing.wav: RIFF (little-endian) data, WAVE audio,
Microsoft PCM, 24 bit, stereo 44100 Hz
$ sox 140867__juskiddink__boing.wav -t wav -r 10000 -c 1 -b 8
140867__juskiddink__boing__v2.wav
$
```

Figura 3: Comprobación de características y conversión del audio para el proyecto:.

Con el uso del *Makefile*, se automatiza el resto del proceso de conversión del formato de los ficheros de audio a un formato de banco de sonidos que aglutina los diferentes sonidos en forma de ondas sin compresión, que es soportado directamente por la NDS. Veamos ahora esta conversión.

## 4.1 De archivos de audio a banco de sonidos

La aplicación **mmutil**[1] permite convertir los ficheros de audio que vamos a utilizar en un formato que la NDS puede leer. Puede trabajar a partir de ficheros de audio en formato de forma de ondas (WAVE) y de audio estructurado (MOD, S3M, XM e IT).

Para ello, cada vez que se modifiquen los ficheros de sonido (*boing.wav* y *bsmusic.mod*, en este caso), el **mmutil** los convertirá (como muestra la Figura 4) a un fichero binario (*soundbank\_bin*) que contiene todos los sonidos que ha encontrado (en el directorio indicado en el proyecto), en formato *raw*. Además, creará los archivos de cabecera *soundbank.h* y *soundbank\_bin.h* en formato de C/C++ para declarar las variables que contienen los sonidos y sus características. Todo ello será enlazado junto con el código, que es una manera de importar los recursos gráficos en ejecutable final.

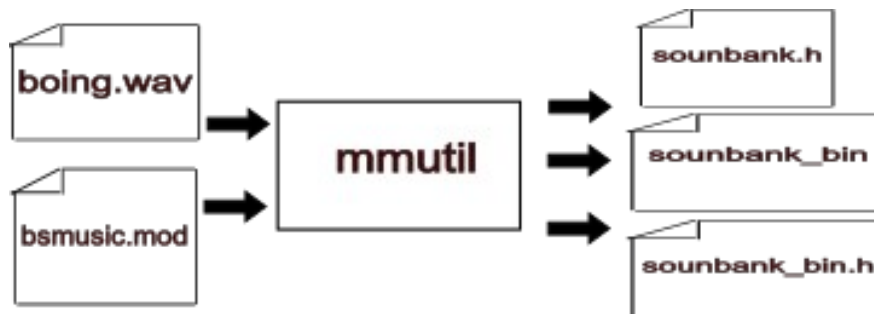


Figura 4: Conversión de ficheros de audio con mmutil.

A partir de los nombres de ficheros de sonido, **mmutil** genera los ficheros de cabecera con el contenido, para nuestro caso, que muestra la Figura 5. Esta figura muestra cómo se invoca a **mmutil** y qué se obtiene tras su ejecución. En ella se puede ver que [1] los ficheros modulares dan lugar a definiciones con prefijo *MOD\_*. Los ficheros de forma de ondas serán nombrados con prefijos *SFX\_*. Y el resto del nombre se obtendrá del nombre del

fichero. En nuestro caso solo tenemos un ejemplo de cada uno, así que vemos un MOD\_BSMUSIC y un SFX\_BOING. Además se crean una serie de definiciones para tener cuenta de cuántos elementos se han incorporado al banco de sonidos (MSL\_NSONGS y MSL\_NSAMPS) y su longitud (MSL\_BANKSIZE). Finalmente vemos los nombres de la estructura de datos que contiene todo el sonido, el banco de sonidos (soundbank\_bin), junto con otros dos vectores que contienen las direcciones, dentro del primero, donde empieza cada elemento sonoro.

```
$ mmutil sonidos/bsmusic.mod sonidos/boing.wav -d -osoundbank.bin -hsoundbank.h
$ cat build/soundbank.h
#define MOD_BSMUSIC 0
#define SFX_BOING 31
#define MSL_NSONGS 1
#define MSL_NSAMPS 32
#define MSL_BANKSIZE 33
$
$ cat build/soundbank_bin.h
extern const u8 soundbank_bin_end[];
extern const u8 soundbank_bin[];
extern const u32 soundbank_bin_size;
$
```

*Figura 5: Ejemplos de conversión de sonidos con mmutil*

## 5 Desarrollo

En la excusa de que está el github, podría rebajar el listado de código d'extensió

```

1.  #include <gfx_ball.h>
2.  #include <gfx_brick.h>
3.  #include <gfx_gradient.h>
4.  #include "ball.h"
5.
6.  ball g_ball;
7.  void resetBall( void );
8.  void setupGraphics();
9.  void updateGraphics();
10. void processLogic( void );
11.
12. // Para el uso del sonido
13. #include "soundbank.h"
14. void inicializarSonidos( void );
15. void liberarRecursosSonidos( void );
16.
17. int main( void ) {
18.     setupGraphics();
19.     resetBall();
20.     inicializarSonidos();
21.
22.     while(1) {
23.         processLogic();           //Actualizacion objetos (moverlos, velocidades, etc)
24.         swiWaitForVBlank();
25.         updateGraphics();         //Pintar los graficos
26.     }
27.     liberarRecursosSonidos();     // Descargar de la memoria los recursos de audio
28. } // Fi de main
...
29.
30. void inicializarSonidos( void ) {
31.     mmInitDefaultMem((mm_addr)soundbank_bin); //Iniciamos Maxmod
32.
33.     mmLoad(MOD_BSMUSIC);           //Cargamos modulo de musica
34.     mmSetModuleVolume( 512 ); // New volume level. Ranges from 0 (silent) to 1024 (normal).
35.
36.     mmLoadEffect( SFX_BOING );     //Cargamos sonido FX
37.     mmEffectVolume( SFX_BOING, 128 ); //Ranges from 0 (silent) to 255 (normal).
38.
39.     mmStart(MOD_BSMUSIC, MM_PLAY_LOOP); //Comenzamos la musica de fondo
40. } // Fi de inicializarSonidos
41.
42. void liberarRecursosSonidos( void ) {
43.     mmStop();
44.     mmUnload(MOD_BSMUSIC);
45.     mmUnloadEffect(SFX_BOING);
46. } // Fi de liberarRecursosSonidos

```

*Listado 1: Contenido nuevo en main.c para poder utilizar el audio.*

```

1.  #include <nds.h>
2.  #include "ball.h"
3.
4.  #include <maxmod9.h>
5.  #include "soundbank.h"
6.
7.  void ballUpdate( ball* b ) { // Físicas
8.      b->x += (b->xvel>>4);    //Gravetat: apply air friction to X velocity
9.      b->xvel = (b->xvel * (256-c_air_friction)) >> 8;    //Gravetat: apply air friction to X velocity
10.     b->xvel = clampint( b->xvel, -max_xvel, max_xvel ); // clamp X velocity to the limits
11.     b->yvel += c_gravity;    // add gravity to Y velocity
12.     b->y += (b->yvel);    // add Y velocity to Y position
13.
14.     if( b->y + c_radius >= c_platform_level ) {    // Bounce on the platform
15.         b->xvel = (b->xvel * (256-c_ground_friction)) >> 8;    // apply ground friction to X velocity
16.         if( b->y > c_platform_level - min_height ) { // the ball has been squished to min. Height?
17.             b->y = c_platform_level - min_height;    // mount Y on platform
18.             // negate Y velocity, also apply the bounce damper
19.             b->yvel = -(b->yvel * (256-c_bounce_damper)) >> 8;
20.             // clamp Y to minimum velocity (minimum after bouncing, so the ball does not settle)
21.             if ( b->yvel > -min_yvel ) b->yvel = -min_yvel;
22.         }
23.         b->height = (c_platform_level - b->y) * 2;    // calculate the height
24.         //mmEffectCancel( SFX_BOING );
25.         handFX = mmEffect( SFX_BOING ); // Hacer sonar para simular el rebote de la pelota
26.         // mmEffectRelease( handFX );
27.     }
28.     else {
29.         b->height = c_diam << 8;
30.     }
31. } // Fi ballUpdate

```

*Listado 2: Contenido nuevo en ball.c para poder utilizar el audio.*

## 6 Conclusiones y cierre

Con este artículo hemos vuelto a poner a disposición de los interesados en el desarrollo de aplicaciones *homebrew* sobre la NDS, un recurso que creemos es de utilidad para servir de apoyo a los que se inician en este complejo contexto de desarrollo de aplicaciones para plataformas de videoconsolas.

Las actualizaciones necesarias del código y la ubicación en un repositorio en *GitHub* son nuestra modesta aportación. Quiero agradecer desde aquí el trabajo de los autores que figuran en la bibliografía y especialmente el de Jose David con quien llevamos adelante el desafío de completar la parte inacaba con la incorporación de la música de fondo y el efecto de audio del rebote de la pelota. Sirva este trabajo como homenaje a estos desarrolladores que han contribuido con su código y con sus explicaciones a que otros puedan adentrarse en este apasionante y difícil campo de desarrollo de aplicaciones para computadores con una arquitectura tan especializada y poco documentada.

Esperamos que el lector se anime a descargar el proyecto desde el *GitHub* [3], leer el original y probar a ejecutar los desarrollos expuestos, p. ej. con el emulador *DeSmuMe*[4] .



Y, por supuesto, experimentar y modificar el código que se publica en el repositorio del proyecto.

## 7 Bibliografía

- [1] Mukunda Johnson. Sitio web. Disponible en <http://mukunda.com/projects.html>.
- [2] Jaén Gomariz, JD. (2015). Tutorial práctico para desarrollo de videojuegos sobre plataforma Nintendo NDS. Disponible en <http://hdl.handle.net/10251/56433>.
- [3] Repositorio del proyecto Escena del tutorial “The bouncing ball”. Disponible en <https://github.com/magusti/NDS-hombrew-development>.
- [4] DeSmuME. Página web del proyecto. Disponible en <http://desmume.org/>.
- [5] J. Amero (Patater). (2008). Introduction to Nintendo DS Programming. Disponible en <https://patater.com/files/projects/manual/manual.html>.
- [6] Boudeville, O. (2008). A guide to homebrew development for the Nintendo DS. Disponible en <http://osdl.sourceforge.net/main/documentation/misc/nintendo-DS/homebrew-guide/HomebrewForDS.html>.
- [7] F. Moya y M. J. Santofimia. (2011). Laboratorio de Estructura de Computadores empleando videoconsolas Nintendo DS. Ed. Bubok Publishing. ISBN. 978-84-9981-039-3.
- [8] Freesound. Disponible en <http://freesound.org/>.
- [9] The Mod Archive . Disponible en <https://modarchive.org/>.
- [10] *Maxmod*. Sitio web disponible en <https://maxmod.devkitpro.org/>.
- [11] Sound eXchange | HomePage. Disponible en <http://sox.sourceforge.net/>
  
- [12] Foxi4. (2012). *DS Programming for Newbies!* Disponible en <https://gbatemp.net/threads/ds-programming-for-newbies.322106/page-6>.
- [13] Grit. *GBA Raster Image Transmogriifier*. Disponible en <http://www.coranac.com/projects/grit> .
- [14] Stair, S. (2006). *GBATEK. Gameboy Advance / Nintendo DS - Technical Info*. Disponible en <https://www.akkkit.org/info/gbatek.htm>.

