UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Introducción al desarrollo con SDL 1.2 para plataforma 3DS. Uso del audio

Apellidos, nombre	**Agustí i Melchor, Manuel** (magusti@disca.upv.es)
Departamento	**Departamento de Informática de Sistemas y Computadores (DISCA)**
Centro	Universitat Politècnica de València

1 Resumen de las ideas clave

Simple DirectMedia Layer (SDL, la Figura 1a muestra el logotipo) [**1**] es una biblioteca de funciones de bajo consumo de recursos, con versiones para la mayoría de lenguajes de programación existentes y distribuida bajo licencia *zlib*. Fue creada por Sam Lantinga en 1998, mientras trabajaba en *Loki Software*, para realizar operaciones de carácter multimedia y multiplataforma. Fue diseñada para proporcionar acceso de bajo nivel a audio, entrada/salida (gestión de eventos producidos por teclado, ratón, *joystick*, etc. y acceso a ficheros), imágenes (en 2D y con soporte hardware para 3D como render de OpenGL), uso de temporizadores y sincronización basada en hilos. SDL está escrita en C, ofrece un nivel común para diferentes sistemas operativos (SO) y conjuga operaciones de bajo nivel con un alto nivel de portabilidad que facilitan el desarrollo multiplataforma, delegando en las interfaces nativas la implementación final de esas operaciones (Figura 1b).

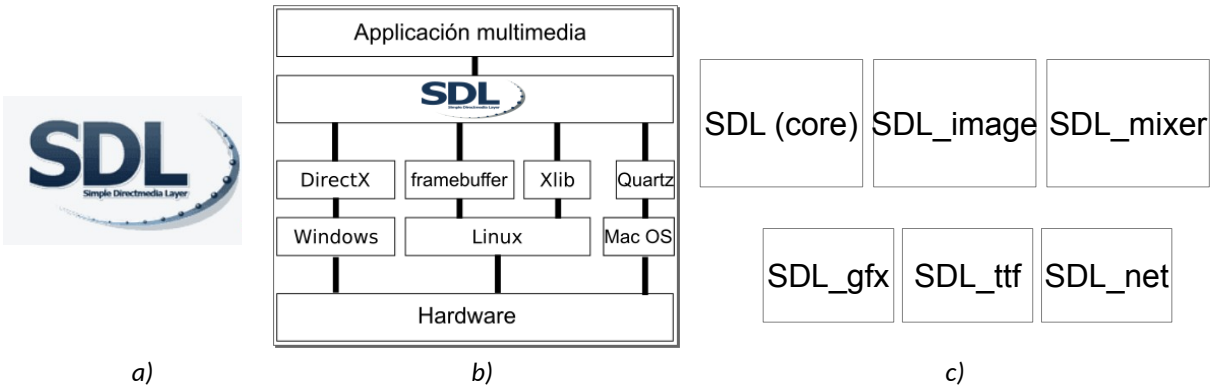


Figura 1: SDL: (a) Logotipo de SDL, (b) Niveles de capa de abstracción del SO sobre el que se ejecuta una aplicación multiplataforma y (c) Módulos de SDL: principal y extensiones oficiales. Imágenes de <https://gbatemp.net/threads/release-sdl-3ds-1-2-15-simple-directmedia-layer-for-3ds.459291/> y <http://easy-learn-computer.blogspot.com/2012/01/learning-sdl-simple-directmedia-layer.html>.

En cuanto a plataformas existentes hoy en día, se puede hablar de tres grupos bastante diferentes:

- El computador de escritorio (véase la Figura 1b) sobre los SO Linux, macOS o Windows (y su correspondientes subsistemas gráficos X11, Quartz o DirectX respectivamente).
- Las videoconsolas, con un SO generalmente propietario y cerrado.
- Los *smartphone*, *bajo Android*, *iOS*, *etc.*.

Entre las plataformas soportadas[1], en la versión oficial, se encuentran las correspondientes al computador de escritorio (bajo Linux, macOS, Windows y otras variantes de Unix). Y también de manera no oficial, esto es realizadas por terceras partes, existen versiones para otras plataformas como videoconsolas y, en particular para **Nintendo 3DS** (N3DS o 3DS para abreviar) y Nintendo ***Switch*** (o, simplemente, *Switch*) entre otras. SDL, actualmente, tiene dos grandes ramas y una tercera en camino[2]: SDL 1.2 (que está presente en el SDK no oficial para 3DS) y SDL 2.0 (que lo está en el de *Switch*).

[1] La lista actual completa se puede ver en “SDL Wiki | Installing SDL” en la URL <https://wiki.libsdl.org/SDL2/Installation>.

En el momento de desarrollar una aplicación de características multimedia, como un videojuego, suele plantearse el tema de su **portabilidad**, esto es, si es posible reconstruir la aplicación en más de una plataforma de computador. SDL ofrece un camino interesante para resolver esta cuestión. El uso de la biblioteca **SDL,** permite realizar una versión portable, con **cambios** mínimos, al ofrecer un nivel de abstracción común por encima de estas plataformas. Para ello, como muestra la Figura 1b, fue diseñada como un API a un nivel de abstracción más alto que el acceso directo a la combinación de hardware y sistema gráfico disponible en cada plataforma. Con el tiempo, nuevas plataformas han sido añadidas a este abanico que puede abordar SDL.

SDL se compone de diferentes módulos (Figura 1c), siendo la "standard library" el **módulo básico** (*core/kernel*) que se encarga de la gestión del modo de vídeo y los eventos de teclado y ratón. Sobre este pueden utilizarse otros módulos (o extensiones oficiales) como *SDL_image* (para soporte de formatos gráficos), *SDL_mixer* (formatos de audio, reproducción y mezcla), *SDL_net* (operaciones de transmisión en red) o *SDL_ttf* (uso de tipos de letra *TrueType*). En este artículo nos vamos a centrar en el soporte de SDL al audio por parte de los módulos **principal (*core) y SDL_mixer sobre la plataforma 3DS*.**

2 Objetivos

Una vez que el lector haya revisado este artículo con detenimiento y explore el código que se adjunta, dispondrá de una referencia para aplicar en el caso de abordar el desarrollo sobre la videoconsola 3DS utilizando la biblioteca SDL. En particular, será capaz de:

- Describir qué es SDL y su situación respecto al uso de audio en la plataforma 3DS con SDL *core*.
- Explorar ejemplos que permitan experimentar con la funcionalidad del módulo *SDL_mixer* de SDL en el contexto de la plataforma 3DS.

Para no extender la longitud del artículo se ha creado un repositorio en GitHub [4] donde se alojarán los ejemplos de código que se referencian es este artículo. El resto de este documento se centrará en describir las características generales de SDL, respecto al uso del audio, referidas a la plataforma 3DS y en explorar ejemplos de uso aplicados a los dos niveles que ofrecen los módulos SDL *core* y *SDL_mixer*.

3 Introducción

La disponibilidad de SDL estable, la encontramos en el conjunto de librerías PORTLIBS distribuidas con *devkitPro* [2] para 3DS en la versión 1.2.15, aunque no incluye el módulo de red (SDL_net). La versión SDL 1.2 está descontinuada oficialmente por parte de SDL, pero se han liberado los fuentes y se sigue manteniendo de forma externa a la propia SDL [4], quienes todavía mantienen la documentación, pero la disponibilidad de ejemplos en su sitio web [1] es muy baja. Como, en particular, sobre la plataforma 3DS, no hay ejemplos portados para poderlos tomar de referencia [3], es interesante disponer de un conjunto de ejemplos que aborden la funcionalidad de SDL 1.2.

[2] Puede leer sobre esta futura versión en el sitio de Github: *Simple DirectMedia Layer (SDL) Version 3.0* <https://github.com/libsdl-org/SDL>.

SDL_mixer complementa el soporte de audio de SDL ofreciendo un mezclador software multicanal. Por defecto[3] es de ocho (modificable en tiempo de ejecución) canales de audio con formato (por defecto) de 16 bits estéreo y un canal de música. Los formatos soportados son FLAC, MP3, Ogg, VOC y WAVE, así como MIDI, MOD y Opus en algunas instalaciones.

Es habitual separar estas dos fuentes de origen de sonido: el audio FX (de efectos) y la música. Los primeros hacen referencia a "clips de audio", que es una manera de referirse a ficheros que contienen grabaciones de corta duración y que están asociados al movimiento de los personajes, disparos, explosiones, choques y otros elementos de interacción entre los componentes de un videojuego que se acompañan de audio para darles un cariz o reforzar su expresividad; por su corta duración suelen disponerse completos en memoria para acceder rápidamente a ellos. El segundo hace referencia a la banda sonora, el audio que acompaña de fondo gran parte o la totalidad del videojuego y, dada su longitud tiene un peso importante, por lo que se busca hacer lo más ligero posible este contenido o reproducirlo en modo *streaming*, esto es, sin tener que cargarlo completamente en memoria.

```
$ /opt/devkitpro/portlibs/3ds/bin/sdl-config --version
1.2.15
$ /opt/devkitpro/portlibs/3ds/bin/sdl-config --cflags
-I/opt/devkitpro/portlibs/3ds/include/SDL -D_GNU_SOURCE=1 -ffunction-sections
-fdata-sections -march=armv6k -mtune=mpcore -mfloat-abi=hard -mword-
relocations -I/opt/devkitpro/libctru/include -DARM11 -D_3DS
$ /opt/devkitpro/portlibs/3ds/bin/sdl-config --libs
-L/opt/devkitpro/portlibs/3ds/lib -march=armv6k -mfloat-abi=hard
-L/opt/devkitpro/portlibs/3ds -lSDL -L/opt/devkitpro/libctru/lib -lcitro3d -
lctru -lm
$
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config sdl --modversion
1.2.15
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config sdl  --cflags
-D_GNU_SOURCE=1 -ffunction-sections -fdata-sections -march=armv6k -
mtune=mpcore -mfloat-abi=hard -mword-relocations -DARM11 -D_3DS
-I/opt/devkitpro/portlibs/3ds/include/SDL -I/opt/devkitpro/libctru/include
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config sdl  --libs
-L/opt/devkitpro/portlibs/3ds/lib -L/opt/devkitpro/portlibs/3ds
-L/opt/devkitpro/libctru/lib -L/opt/devkitpro/portlibs/3ds
-L/opt/devkitpro/libctru/lib -march=armv6k -mfloat-abi=hard -lSDL -lcitro3d -
lctru -march=armv6k -mfloat-abi=hard -lSDL -lcitro3d -lctru -lm
$
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config SDL_mixer  --
modversion
1.2.12
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config SDL_mixer  --
cflags
-D_GNU_SOURCE=1 -ffunction-sections -fdata-sections -march=armv6k -
mtune=mpcore -mfloat-abi=hard -mword-relocations -DARM11 -D_3DS
-I/opt/devkitpro/portlibs/3ds/include/SDL
-I/opt/devkitpro/portlibs/3ds/include
-I/opt/devkitpro/portlibs/3ds/include/SDL -I/opt/devkitpro/libctru/include
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config SDL_mixer  --libs
-L/opt/devkitpro/portlibs/3ds/lib -L/opt/devkitpro/portlibs/3ds
-L/opt/devkitpro/libctru/lib -L/opt/devkitpro/portlibs/3ds
-L/opt/devkitpro/libctru/lib -lSDL_mixer -lmad -lvorbisidec -logg -lmikmod -
lm -march=armv6k -mfloat-abi=hard -lSDL -lcitro3d -lctru -march=armv6k -
mfloat-abi=hard -lSDL -lcitro3d -lctru -lm
```

Listado 1: Dependencias de SDL con sdl-config y pkg-config.

[3] Se puede ver en el fichero *$DEVKITPRO/portlibs/3ds/include/SDL/SDL_mixer.h*.

Las herramientas disponibles en *devkitPro* (*sdl-config* y *pkg-config*) nos permiten caracterizar las dependencias de SDL y *SDL_mixer* en este contexto, véase el Listado 1. Utilizaremos esta información cuando hablemos de las dependencias de un proyecto con SDL 1.2 en 3DS para generar el distribuible final, para lo que necesitaremos saber cuáles son y cómo indicarlas.

4 Desarrollo

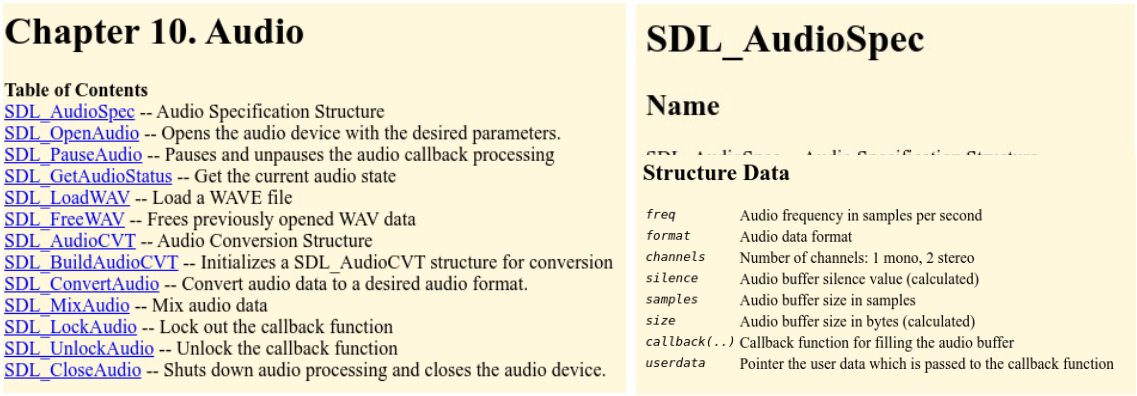
Veamos ahora ejemplos de código comentados sobre el uso de operaciones relativas al audio. Distinguiremos dos niveles:

- Las operaciones más básicas y de más bajo nivel que implementa el módulo SDL (*core*).
- Las operaciones de más alto nivel que ofrece el módulo *SDL_mixer*.

4.1 Ejemplo de uso del audio en SDL *core*

El soporte de audio en SDL está basado, como es habitual en todos los computadores actuales, en la gestión de audio en formato digital, digitalizando o convirtiendo de analógico a digital la representación de una señal de sonido.

Esta conversión representa mediante *samples* (muestras para uno o más canales) el sonido en un instante de tiempo, esto es, el muestreo de una señal que en origen (en el mundo real) es analógica y que ha sido convertida a un formato digital procesable por un computador. Además, es reversible, esto est, que es posible volver a obtener una señal analógica para que la podamos oír: si procedemos a “alimentar” al hardware de audio con muestras, este hará el proceso de conversión de digital a analógico



a) *b)*

Figura 2: Soporte al uso de audio en SDL (core): (a) operaciones y (b) estructuras de datos.

Esto se concreta en una serie de operaciones básicas, véase Figura 2a, que nos permite señalar que la gestión de audio en el módulo básico de SDL (en esta versión 1.2.15) está basada en un pequeño (once) conjunto de operaciones y dos estructuras de datos.

De las operaciones, Figura 2a, destacaremos que está basada en:

```
1.  #include <3ds.h>
2.  #include <stdio.h>
3.  #include <SDL/SDL.h>
4.
5.  void audioCallback(void* userdata, Uint8* stream, int len) {
6.      SDL_RWops* rw = (SDL_RWops*)userdata;
7.      SDL_RWread(rw, stream, 1, len);
8.  }
9.
10. int main(int argc, char *argv[]) {
11.     SDL_Surface *screen;
12.     SDL_AudioSpec wavSpec;
13.     Uint8* wavBuffer;
14.     Uint32 wavLength;
15.
16.     SDL_Init(SDL_INIT_AUDIO);
17.     screen = SDL_SetVideoMode(400, 240, 32,
18.                     SDL_SWSURFACE | SDL_TOPSCR | SDL_CONSOLEBOTTOM);
19.     romfsInit();
20.
21.     if (SDL_LoadWAV("romfs:/test.wav", &wavSpec, &wavBuffer,
22.                     &wavLength) == NULL) {
23.       printf("Error en SDL_LoadWAV: %s\n","romfs:/test.wav");
24.     }
25.     else
26.       printf("SDL_LoadWAV: %s --> frec %d, canales %d y muestras %d\n",
27.              "romfs:/test.wav", wavSpec.freq, wavSpec.channels,
28.              wavSpec.samples );
29.
30.     SDL_AudioSpec obtainedSpec;
31.     SDL_AudioSpec desiredSpec;
32.     desiredSpec.freq = wavSpec.freq;
33.     desiredSpec.format = wavSpec.format;
34.     desiredSpec.channels = wavSpec.channels;
35.     desiredSpec.samples = 2048;
36.     desiredSpec.callback = audioCallback;
37.     desiredSpec.userdata = SDL_RWFromConstMem(wavBuffer, wavLength);
38.
39.     SDL_OpenAudio(&desiredSpec, &obtainedSpec);
40.     SDL_PauseAudio(0);
41.     printf("Espere un poc");
42.     SDL_Delay(10000);
43.
44.     SDL_FreeWAV(wavBuffer);
45.     SDL_CloseAudio();
46.     SDL_Quit();
47.
48.     return 0;
49. }
```

Listado 2: Ejemplo de SDL 1.2 básico con SDL (core).

- El acceso al hardware de audio (las operaciones de *Open/Close*, *Pause*, *GetAudioStatus*, *MixAudio* y *Lock/Unlock*).

- Un único formato de audio digital[4] (*WAVE*).
- Y dos estructuras de datos, *AudioSpec* y *AudioCVT*.
  La primera permite configurar el hardware de audio y, la segunda, consultar sus características, véase Figura 2b, como frecuencia, formato de los valores digitales, canales, tamaño de un *buffer* (en número de muestras y en número de bytes) y una o varias funciones de *callback* que actualizan el contenido de un *buffer*.

Del estudio de esta documentación debemos recalcar que:

- Solo se pueden mezclar dos *buffers* de audio (*SDL_MixAudio*).
- Se puede configurar el hardware con unos valores (*SDL_AudioSpec*) y convertir (*SDL_AudioCVT*) el audio a ese formato para que se ajuste a las posibilidades del subsistema de audio.
- La carga de un fichero de audio (*SDL_Load WAV*) genera un *buffer* con todo el sonido extraído del archivo, aunque es posible ir generando o cargando trozos del archivos sobre la marcha, para aligerar la necesidad de recursos. En este caso las funciones de *callback* cobran especial importancia y suponen una tarea compleja[5] que ha de asumir el desarrollador.

El Listado 2 muestra un ejemplo básico de uso de audio mediante el módulo SDL *core* en el que podemos ver que, línea 16, es necesario inicializar el componente de audio de SDL. Se puede cargar el contenido de un fichero, línea 21 y 22, lo que nos permitirá mostrar (o disponer) de información del audio cargado. Con ello se puede configurar el hardware de audio, si no se tiene intención de cargar otro audio, claro. Las líneas 30 a la 37 preparan la estructura de datos de configuración deseada y con ella inicializan el dispositivo, línea 39, lo que podría convertirse en un "diálogo" porque este contesta con una lista de valores "obtenidos" que podrían ser diferentes.

A partir de ese momento, la línea 40 empieza a reproducir el audio (o lo para) lo que traduce en invocar (o dejar de hacerlo, respectivamente) a las funciones de *callback*. Y, tras esperar (línea 42) un tiempo, se continúa la ejecución, liberando los recursos (líneas 44 a 46) asignados.

```
        ...
# CFLAGS    +=    $(INCLUDE) -D__3DS__
CFLAGS      +=    $(INCLUDE) -D__3DS__ `pkg-config sdl --cflags`
        ...
# LIBS      := -lctru -lm
LIBS  := `pkg-config sdl --libs` -lctru -lm
        ...
```

Listado 3: Modificaciones en el Makefile para el ejemplo SDL 1.2 básico con SDL (core).

El Listado 3 muestra las variaciones que ha sido necesario introducir en el fichero *Makefile* para obtener el ejecutable final. Por brevedad en la exposición, solo se han incluido las líneas modificadas, indicando la línea original y la línea con los cambios realizados a continuación. En ambos casos hacen referencia a las dependencias del módulo SDL (*core*) que se utiliza en este código y que hemos expuesto en el Listado 1 del apartado 3 Introducción.

[4] El formato *Waveform Audio File Format* (WAVE o WAV), fue desarrollado por IBM y Microsoft en 1991. Puede consultar más al respecto en <https://en.wikipedia.org/wiki/WAV>.

[5] En *Audio Examples* <https://www.libsdl.org/release/SDL-1.2.15/docs/html/guideaudioexamples.html> podrá encontrar la sugerencia para desarrollar este concepto.

4.2 Ejemplo de uso de audio con el módulo *SDL_mixer*

Este módulo ofrece facilidades para la gestión de audio implementadas sobre la base de SDL *core*. Si examinamos el contenido de SDL_mixer.h[6], del que recalcaremos que:

- Es el que ofrece soporte a formatos diferentes de WAVE como FLAC, MOD, MP3 u OGG (y lo hace contando con que hay instaladas unas bibliotecas específicas de esos formatos).
- De partida, se definen ocho canales (y se proporcionan las funciones para hacer variar este número hacia arriba y también hacia abajo) y un formato por defecto para el audio que pueden alojar que, por supuesto, se puede modificar en tiempo de ejecución.

Se observará que para conseguir la funcionalidad del apartado anterior, ahora son necesarias menos líneas y el nivel de abstracción ha crecido. Podemos observarlo en las líneas 13 y 14 que se encargan de cargar el fichero (en formato WAVE) para facilitar la comparación con el apartado anterior y de empezar a reproducirlo.

```
1.  #include <3ds.h>
2.  #include <stdio.h>
3.  #include <SDL/SDL.h>
4.  #include <SDL/SDL_mixer.h>
5.
6.  int main(int argc, char *argv[]) {
7.      SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO);
8.
9.      Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 2048);
10.
11.     romfsInit();
12.
13.     Mix_Chunk *sound = Mix_LoadWAV("romfs:/test.wav"); //sound.wav");
14.     Mix_PlayChannel(-1, sound, 0);
15.
16.     while (Mix_Playing(-1)) {
17.         SDL_Delay(1000);
18.     }
19.
20.     Mix_FreeChunk(sound);
21.     Mix_CloseAudio();
22.     SDL_Quit();
23.
24.     return 0;
25. }
```

Listado 4: Ejemplo de SDL 1.2 básico con SDL_mixer.

Ahora podemos preguntar por el estado de reproducción de un *buffer*, *línea 16*, y esperar (línea 17) a que termine sin necesidad de fijar ese valor de espera en el código para cada fichero de audio manualmente. Por último, entre las líneas 20 y 21 se procede a liberar los recursos asignados tanto de memoria ocupada por el audio, como de uso del hardware de audio.

[6] O bien en nuestra instalación local (en $DEVKITPRO/portlibs/3ds/include/SDL/SDL_mixer.h) o en la URL <https://www.libsdl.org/projects/SDL_mixer/release/SDL_mixer-1.0.6/SDL_mixer.h>.

Como se utiliza ahora el módulo *SDL_mixer*, hay que proceder a añadir las dependencias de este en el fichero *Makefile* que permite construir el ejecutable final. El Listado 5 muestra las variaciones que ha sido necesario introducir en el fichero *Makefile* para obtener el ejecutable final. Como en el Listado 5, solo se han incluido las líneas modificadas, indicando la línea original y la línea con los cambios realizados a continuación.

```
        ...
# CFLAGS    +=    $(INCLUDE) -D__3DS__
CFLAGS      +=    $(INCLUDE) -D__3DS__ `pkg-config sdl --cflags`
        ...
# LIBS      := -lctru -lm
LIBS  := -lSDL_mixer -lmikmod  -lvorbisidec -logg -lmad `pkg-config sdl --
libs` -lctru -lm
        ...
```

Listado 5: Modificaciones en el Makefile para el ejemplo SDL 1.2 básico con SDL_mixer.

Observe que ahora se incluyen también referencias a otras bibliotecas de funciones sobre las que se apoya *SDL_mixer* para ofrecer el acceso a diferentes formatos de ficheros de audio como MOD, Vorbis, OGG o MP3. Recuerde que hemos visto ya cómo obtener estas dependencias para *SDL_mixer* anteriormente, véase el Listado 1 del apartado 3 Introducción.

5 Conclusión y cierre

A lo largo de este objeto de aprendizaje hemos visto cómo se gestiona el audio en SDL desde sus operaciones básicas en el módulo *core* de SDL hasta las operaciones más complejas que ofrece el módulo SDL_mixer. Por supuesto que hay más operaciones relacionadas con el audio, si tiene la curiosidad además de revisar la documentación, puede consultar el código de los ejemplos de la documentación oficial[7]: *PlayWave* y *PlayMusic* sobre el uso de SDL_mixer.

No hemos insistido en el interfaz visual para centrar la atención en las operaciones relacionadas con el audio. Sería interesante, ahora volver sobre los ejemplos para crear una interfaz que permita utilizar los controles de la videoconsola para subir o bajar el volumen, cambiar de sonidos en ejecución y mostrar la ocupación de los canales en un momento dado.

Ahora, con estas reflexiones es cuestión de proponerse un ejemplo propio. ¿Qué te parece si aprovechamos las opciones básicas exploradas para hacer un juego estilo *Simon*[8] o un pequeño instrumento como un teclado o una batería[9]? ¿Te te animas, estimado lector?

[7] Disponibles en <https://github.com/libsdl-org/SDL_mixer/blob/main/examples/>.

[8] Puede leer sobre la historia de este legendario juego en <https://es.wikipedia.org/wiki/Simon_(juego)>.

[9] Como en https://www.youtube.com/watch?v=kPzu7vwfnCs, https://www.youtube.com/watch?v=sgoOT20EQDg, https://recursivearts.com/virtual-piano/ o https://www.virtualdrumming.com/.

6 Bibliografía y referencias

[1] SDL. Sitio web. <https://www.libsdl.org/>.

[2] devkitPro / 3DS examples. <https://github.com/devkitPro/3ds-examples>.

[3] Documentación de SDL 1.2.15 <https://www.libsdl.org/release/SDL-1.2.15/docs/html/guidevideo.html>.

[4] Repositorio de los ejemplos de este artículo <https://github.com/magusti/3DS/3DS_SDL_1_2_introAudio>.

[5] Wikipedia - Simple DirectMedia Layer. Disponible en <https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer>.