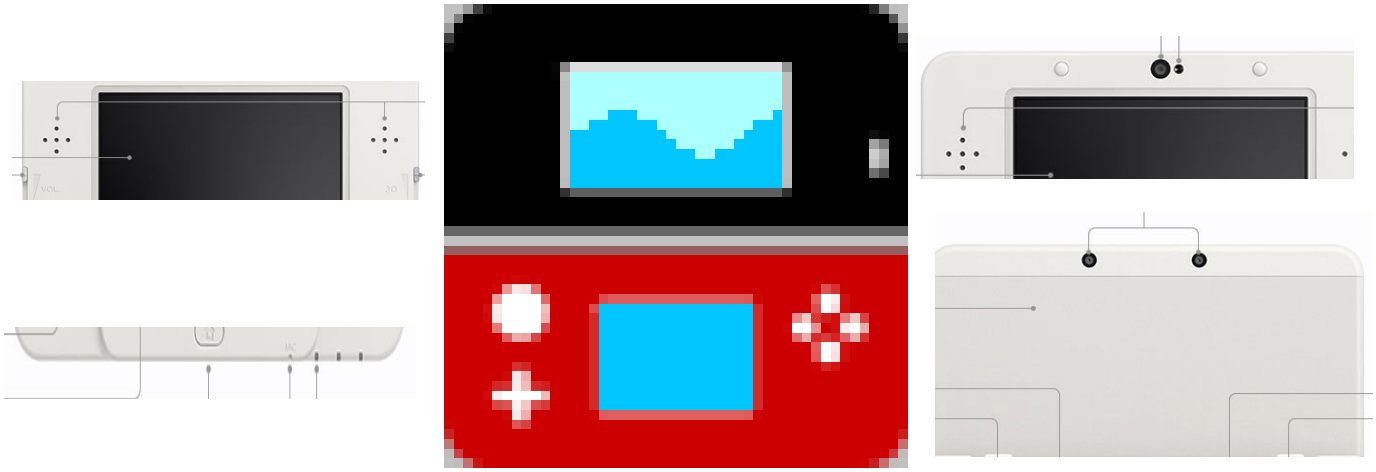
Arquitectura y Entornos de desarrollo para Videoconsolas

Práctica 4
Uso del audio y captura de imágenes en la videoconsola 3DS



M. Agustí
Marzo 2021

La imagen central de la portada ha sido extraída de un icono de la instalación de *DevkitPro*.

La práctica se asume que va a realizar como si fuera en el entorno GNU/Linux que tenemos disponible en el laboratorio, pero es perfectamente realizable en otros sistemas operativos. Vamos a abordar los servicios [1] que ofrece el sistema para el acceso al *hardware* de audio y la cámara. Esto supone, por un lado, la configuración de su emulación en *Citra* y, por el otro, el estudio de la parte del API para acceder a estas funcionalidades.

Es muy recomendable tener a mano la documentación sobre *libctru* que se describió cómo generar en el "Anexo I Documentación en local de libctru, citro3d y citro2d" de la práctica 2 "Acceso a información en el hardware de la videoconsola 3DS".

Recuerde que ha de guardar los resultados de sus acciones a lo largo de las prácticas para confeccionar el porfolio de prácticas, el apartado de trabajo autónomo detalla la realización de esta práctica a entregar. No descuide hacer copias de lo que necesite del laboratorio en su espacio del servidor de la asignatura.

1 Introducción

Abordamos en esta práctica los servicios que ofrece esta plataforma de cara a dar soporte a las capacidades *hardware* de la consola. Hablaremos de los ejemplos bajo el directorio *audio* y *camera*, que hacen uso del DSP que incorpora la videoconsola para el tratamiento de audio y de la cámara para obtener imagen o secuencia de imágenes. Para ello exploraremos los ejemplos de código disponibles en la documentación del SDK de *devKitPro* que habrá copiado en su espacio de usuario y la documentación disponible [2]. Las rutas a los ejemplos que aquí se indican asumen que se conoce el directorio base en que cada uno dejó esta copia de los ejemplos y se indican con la ruta relativa al mismo.

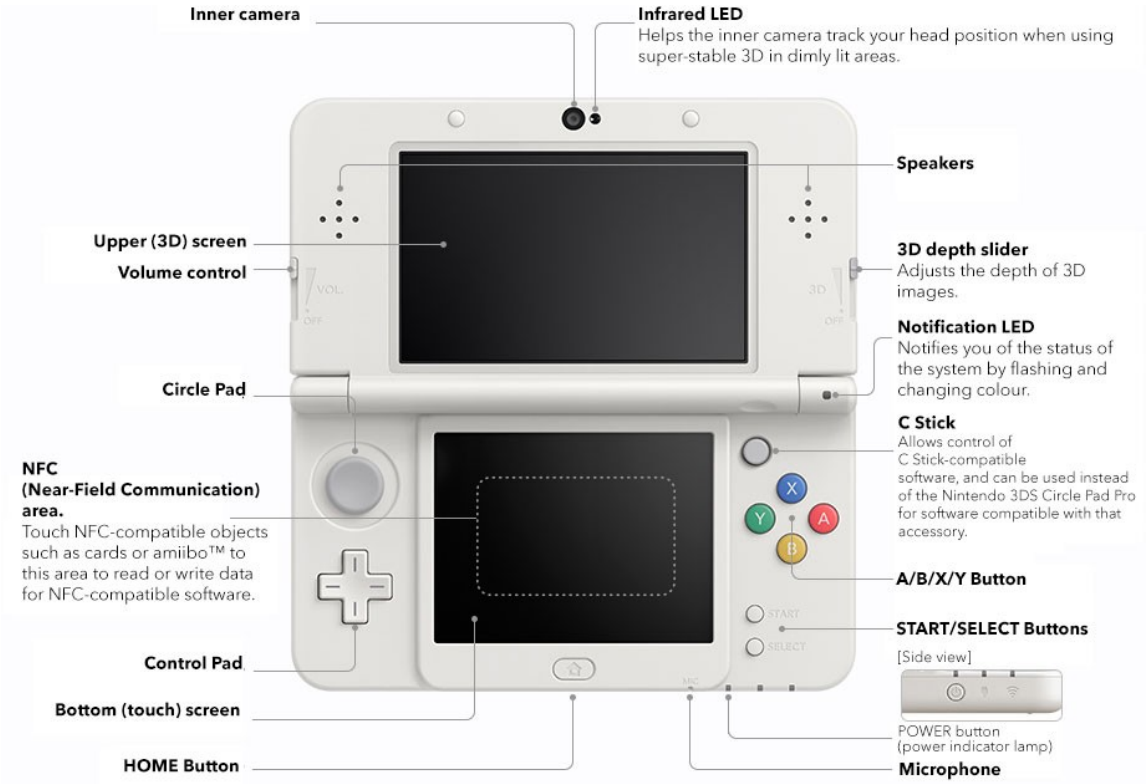


Figura 1: Detalles de los elementos del interfaz de la New Nintedo 3DS: observe la posición de la "Inner camera" y el "Infrared LED" (imagen obtenida de <https://www.giantbomb.com/images/1300-2686369>).

Aunque fuera de nuestros objetivos en esta práctica, cabe mencionar:

- y de las opciones de reproducción de vídeo en formato H264, del ejemplo mvd
- Del acceso al micrófono, con el ejemplo audio/mic.
- el ejemplo de *qtm*[1], que está relacionado con el uso de la imagen en forma de adquisición de secuencia de imágenes. Este servicio está encargado de obtener el posicionamiento y seguimiento de la cabeza para adaptar la función estereoscópica de la pantalla superior al punto de vista del usuario. Véase la pareja de cámara y emisor de led infrarrojo que hay en la parte superior frontal de la consola de la Figura 1.

2 Audio

El corazón de la gestión del audio en la 3DS es un DSP (*Digital Signal Processor*): el CEVA TeakLite[2], que funciona a 134Mhz, con 24 canales disponibles y que trabaja a una frecuencia de muestreo de 32.728Hz. Las aplicaciones y juegos que ejecuta la 3DS, por tanto sobre la CPU de la misma, se comunican con el DSP [3] para reproducir el audio. Esto se hace tanto a través del servicio `dsp::DSP`, como a través de una zona de memoria compartida:

- El servicio `dsp::DSP` ofrece operaciones para inicialización del DSP, incluyendo la carga de *firmware*.
- El área de memoria compartida se utiliza para agilizar la comunicación entre la aplicación en la CPU y el *firmware* en el DSP.

Para emular la operativa del audio en *Citra*, se ha venido desarrollando una emulación de alto nivel (*High Level Emulation*[3] o HLE por sus siglas), buscando huir de la necesidad de disponer de una versión de ese *firmware* propietario al que solo puede acceder el propietario de una consola física y utilizando utilidades que son tildadas, en muchos casos, de ser muy cercanas a los mecanismos que se utilizan para la piratería. Si se dispusiera de la especificación completa del contenido de ese *firmware* se podría implemente una emulación del audio que se conoce como de bajo nivel (*Low Level Emulation* o LLE).

Para realizar la experimentación de este apartado en el laboratorio es necesario:

- Configurar del uso del audio en el emulador *Citra*: vease Anexo I – Configuración de Citra. Será necesario realizar las instrucciones que allí figuran solo una vez, pero recuerde que ha de hacerlo en cada equipo donde vaya a utilizar *Citra*. Básicamente esto es debido a que el DSP incorpora los "códecs" para formatos como WAVE, MP3, AC3 o AAC-HLE y que, en el entorno emulado deben ser realizados por software en el sistema operativo donde se ejecute *Citra*.
- Explorar el API que proporciona *devKitPro*. Esto lo veremos a partir de los ejemplos que acompañan al SDK [1].

Comprobemos que podemos utilizar el sonido en nuestro emulador: esto es, que *Citra* está configurado para usar el audio. No lo podemos hacer desde la versión que ejecutamos con la orden citra[4], así que lo haremos desde la versión de *Qt*, la que ejecutamos al escribir la orden *citra-qt*. Desde el menú *Emulation* | *Configure* cambiaremos la configuración de *Citra* y la modificación realizada aquí quedará en la configuración común de ambas versiones y, por lo tanto, también es

1 Más información al respecto en "QTM Services - 3dbrew*"* <https://www.3dbrew.org/wiki/QTM_Services >.
2 Fuente: 3dsbrew.org <https://www.3dbrew.org/wiki/Hardware>.
3 Más información al respecto en <https://emulation.gametechwiki.com/index.php?title=High/Low_level_emulation&oldid=30150>.
4 La que está implementada con SDL2 .

utilizada por la otra versión de *Citra*.

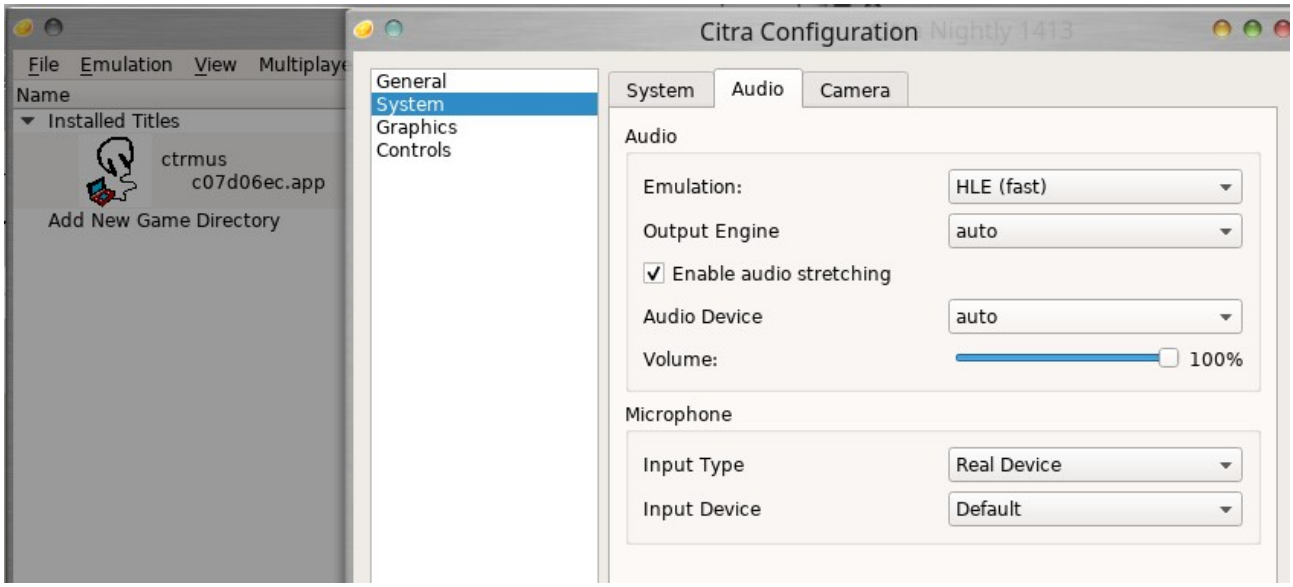


Figura 2: Configuración del audio en Citra, en la versión citra-qt.

La Figura 2 muestra el diálogo de configuración de la pestaña *Audio* del panel *System* de *Citra*, en la que observamos que hay emulación de salida y de entrada de audio disponible.

La operativa relativa al audio en el API, la vamos a desarrollar a partir de los ejemplos del SDK y está dividida en tres partes que pertenecen a la librería *libctru*:

- Las operaciones con prefijo "DSP_", que corresponden a la gestión de la comunicación entre la aplicación (el proceso que se ejecuta en la CPU en la memoria central) y el proceso que ejecuta el DSP y que utiliza la memoria (*caché*) del DSP.
- Las operaciones con prefijo "CSND_" que permiten ejecutar operaciones de bajo nivel de generación de señales básicas de sonido y que están siendo reemplazadas por el siguiente módulo.
- Las operaciones con prefijo "NDSP_" que permiten ejecutar operaciones de gestión de los canales, mezclador, volumen, …, audio envolvente o el uso del micrófono.

2.1 Síntesis por generación de tonos

Veamos ahora como generar señales de sonido básicas y cómo utilizar el "doble buffer" del DSP para tener sonido reproduciéndose mientras se generan nuevas muestras de sonido. Para ello vamos a empezar nuestra exploración con el ejemplo del SDK para la 3DS que está en el directorio *audio/streaming*. Este ejemplo muestra la inicialización del módulo NDSP, la configuración de algunos parámetros de la reproducción del sonido y la generación de las muestras de sonido a partir de una formulación de una señal senoidal

Ejercicio 1. Revise el código del ejemplo mencionado, localice las funciones que llevan prefijo "DSP_" o "NDSP_" y copie sus cabeceras de la documentación propia de *libctru* [2].

Ejercicio 2. El código del ejemplo *streaming* utiliza valores de 32 bits en dos partes de 16 para rellenar un *buffer* que después hará sonar. Compruebe para qué se utiliza cada componente escuchando, con los aurículares, la versión original y la que resulta de modificar la línea que dice

```
dest[i] = (sample<<16) | (sample & 0xffff);
```

por

```
dest[i] = (sample<<16) | 0x0000;
```

Ejercicio 3. El código del ejemplo *streaming* utiliza un mezclador de doce elementos de los que dos los pone a valor 1.0 y el resto a 0.0 Compruebe qué hacen esos valores comparando la versión original del código de *streaming* con el que resulta de modificar la línea que dice

mix[0] = 1.0;

por valores más pequeños. como p. ej.:

mix[0] = 0.5;

Ejercicio 4. Implemente, a partir del ejemplo *audio/streaming,* una versión que permita escuchar los tonos entre 20Hz y 20KHz, variando en intervalos de 100Hz.

2.2 Procesado de señal

Veamos ahora cómo el DSP se encarga de procesar la señal de sonido en tiempo real. Para ello vamos a estudiar el ejemplo del SDK para la 3DS que está en el directorio *audio/filters*. Este ejemplo muestra la inicialización del módulo NDSP, la configuración de algunos parámetros de la reproducción del sonido y la selección de un filtro u otro en función de la elección del usuario que es mostrada en pantalla con mensajes de texto.

Ejercicio 5: Revise el código del ejemplo mencionado, localice las funciones que llevan prefijo "ndspChnIirBiquadSet" y copie sus cabeceras de la documentación propia de *libctru* [2].

2.3 Uso del micrófono

Veamos ahora cómo el CSND se encarga de procesar la señal de sonido en tiempo real. Para ello vamos a estudiar el ejemplo del SDK para la 3DS que está en el directorio *audio/mic*. Este ejemplo muestra la inicialización del módulo CSND, la configuración de algunos parámetros de modo de funcionamiento del servicio MICU y la captura de las muestras de sonido.

Deberá comprobar en la configuración de citra-qt que está asignado algún dispositivo de entrada de audio para el uso de este ejemplo, como se muestra en la Figura 3. En caso de no disponer, puede cambiar el desplegable de "Tipo de Entrada" al valor "Ruido Estático". Tenga en cuenta que la lista de dispositivos que aparece en el desplegable de "Dispositivo de Entrada" dependerá de los disponibles en su equipo y no tiene porqué coincidir con los de la figura.

Ejercicio 6: Compruebe que en su equipo se puede utilizar el ejemplo *audio/mic* (tanto si lo escucha reproducido por el propio código o no) guardando los datos en bruto (*raw* o PCM) directamente en binario en un archivo en la tarjeta de memoria emulada e importándolos en un editor de formas de onda.

Para ello, inserte el siguiente código

```
FILE *file = fopen("microtest.bin","wb");
if (file == NULL) break;
fwrite(audiobuf, 1, audiobuf_pos, file);
fclose(file);
```

que creará un archivo "microtest.bin" en[5]

5 En la configuración de Citra para Linux.

~/.local/share/citra-emu/sdmc/microtest.bin

justo después de la sentencia

```
printf("Starting audio playback...\n");
```

Y, por ejemplo en Audacity, se puede importar el resultado obtenido con la opción de menú "Archivo | Importar | Datos en bruto …" y seleccionando los parámetros que muestra la Figura 4.

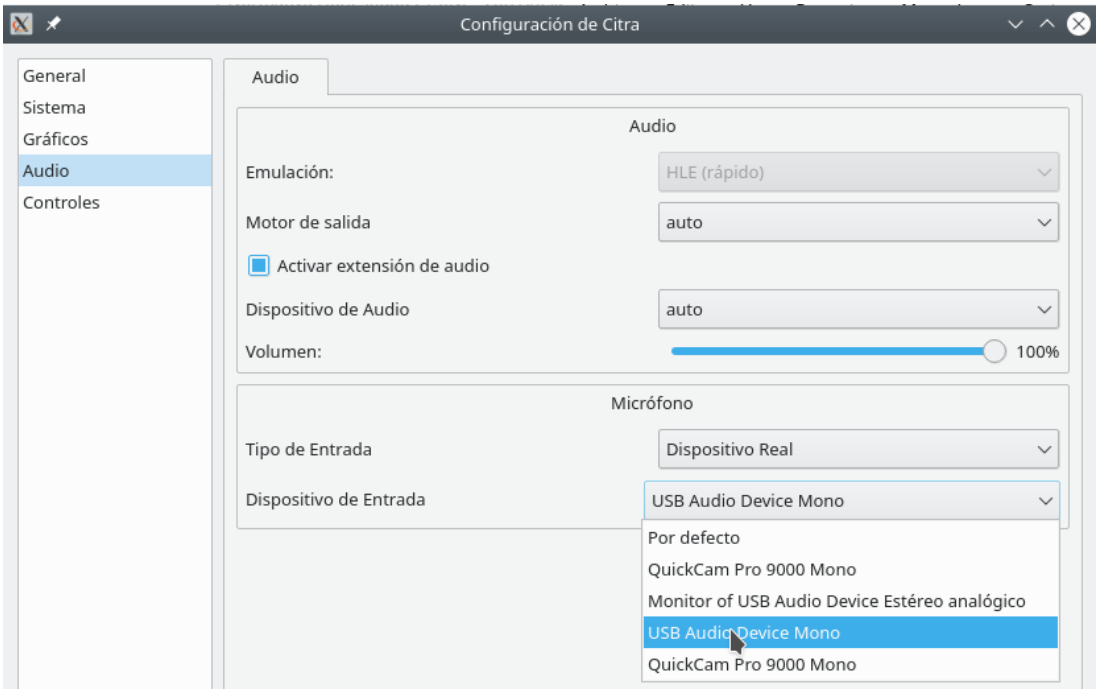


Figura 3: Configurar Citra para usar un dispositivo de entrada de audio real seleccionado por el usuario o uno simulado (Tipo de Entrada = Ruido Estático).

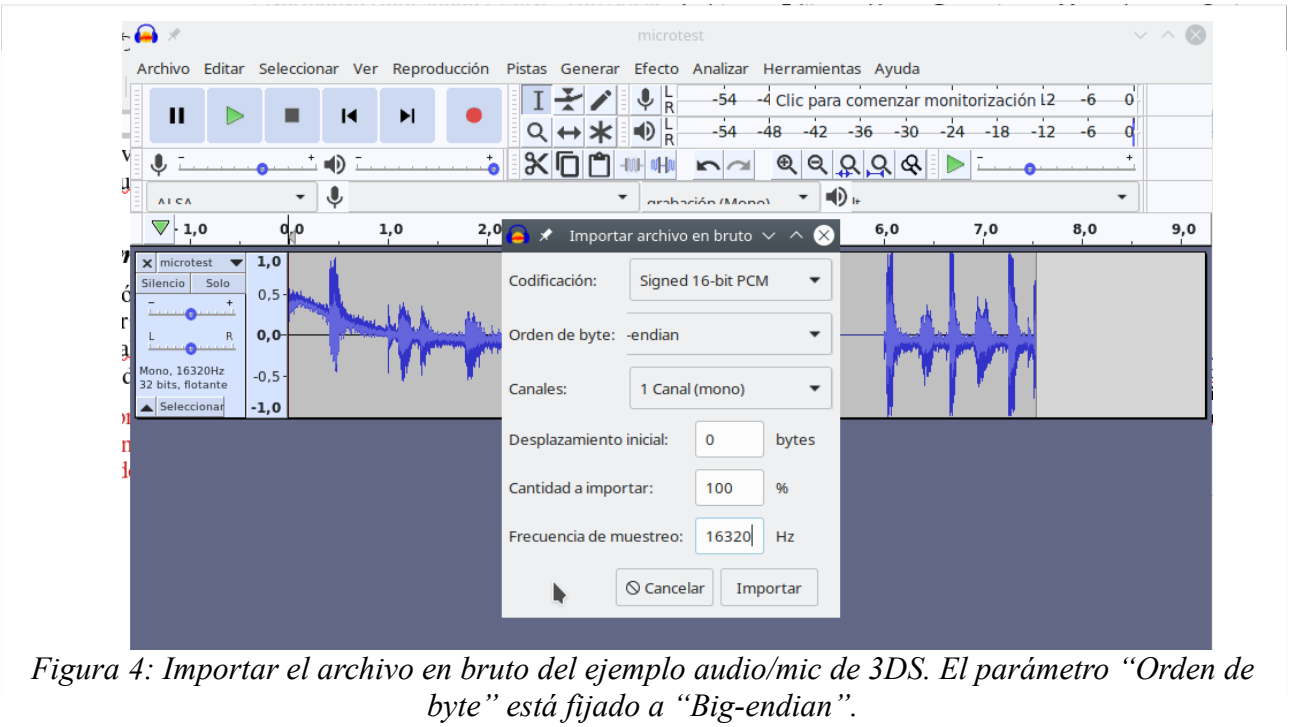


Figura 4: Importar el archivo en bruto del ejemplo audio/mic de 3DS. El parámetro "Orden de byte" está fijado a "Big-endian".

2.4 Uso de ficheros en formato Opus

Veamos ahora cómo el DSP se encarga de reproducir el contenido de un fichero de audio en un formato no nativo del de plataforma 3DS. Para ello vamos a estudiar el ejemplo del SDK para la 3DS que está en el directorio *audio/opus-deconding*. Este ejemplo muestra el uso de la librería libopusfile que forma parte de las PORTLIBS de DevkitPro para 3DS[6], por lo que este ejemplo también es de interés para ver cómo poder utilizar otras de las librerías de este grupo. Esta librería es la encargada de leer el fichero en formato Opus y sacar de él las muestras de sonido en PCM que contiene. Estas serán llevadas al DSP en bloques de tamaño del *buffer* declarado en el código (WAVEBUF_SIZE).

Revise el código del ejemplo mencionado, localice en el código del programa principal:

- Las funciones con prefijo *ndsp* que programan al DSP para que vaya reproduciendo el sonido que se va trayendo sobre uno de los 3 *buffers* (*s_waveBufs*) que se emplean a modo de lista circular y en la que hay que sincronizar el acceso del DSP para leer datos (en plano) que reproducir, con el acceso en paralelo del hilo que lee (y decodifica) el archivo de audio, depositando el audio en esas áreas de memoria.
- Las inicializaciones de las funciones de uso de hilos (*threadCreate, threadJoin y threadFree*).
- Las funciones que sirven de operaciones de sincronización al estilo de las operaciones con semáforos (*LightEvent_Wait y LightEvent_Signal*).

Ejercicio 7: Busque[7] un ejemplo de sonido de banda sonora y conviértalo[8] a Opus para comprobar que el código es capaz de reproducir el fichero que se encuentra en el directorio *romfs* en el momento de la compilación del ejecutable.

3 Cámara

Este apartado se basa en la experimentación sobre los ejemplos de uso de la cámara del SDK que están dentro del directorio *3ds-examples-master/camera*, que son *image* y *video* y que hacen referencia a la toma de imágenes estáticas y a la de secuencias de imágenes en vivo, respectivamente.

En primer lugar trataremos el ejemplo de *3ds-examples-master/camera/image* que es el que describe cómo se toman las imágenes (instantáneas) a través de la librería *libctru*. Este ejemplo permite tomar una nueva fotografía utilizando la cámara frontal (Figura 1) o las traseras que se muestran en la Figura 5 como “Outer cameras”, al pulsar el botón R (*R button* en la Figura 5).

6 Se pueden encontrar en ${DEVKITPRO}/portlibs/3ds/lib/

7 Ejemplos de sonidos para videojuegos pueden encontrarse , p. ej. en el sitio web de Mark Sparling <http://marksparling.ca/> o en “19 Best Websites to Download Royalty-Free Music For Games” <https://blog.felgo.com/game-resources/free-music-for-games>.

8 Para ver de convertir los archivos de audio, en caso de ser necesario, puede consultar “How to convert a sound file to Opus” <https://askubuntu.com/questions/211054/how-to-convert-a-sound-file-to-opus>.

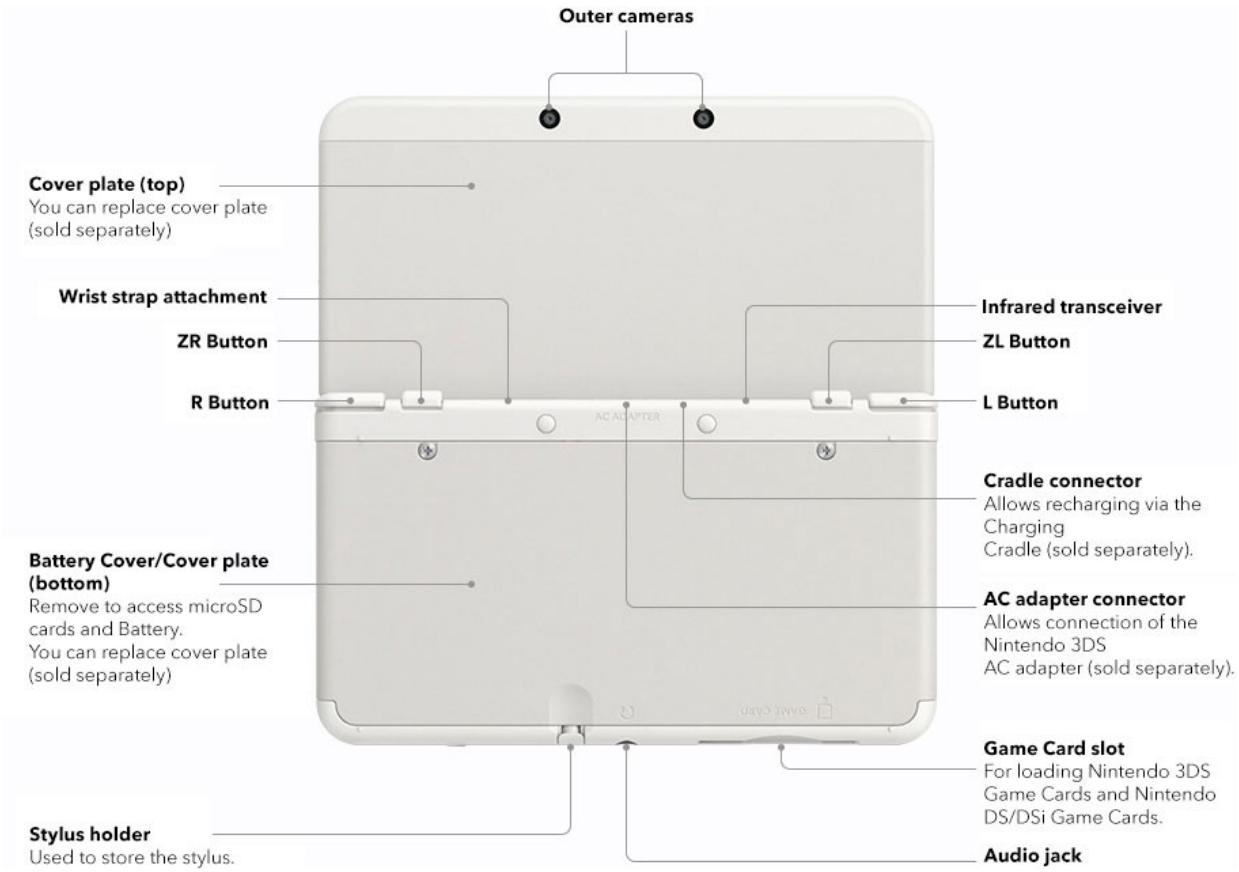
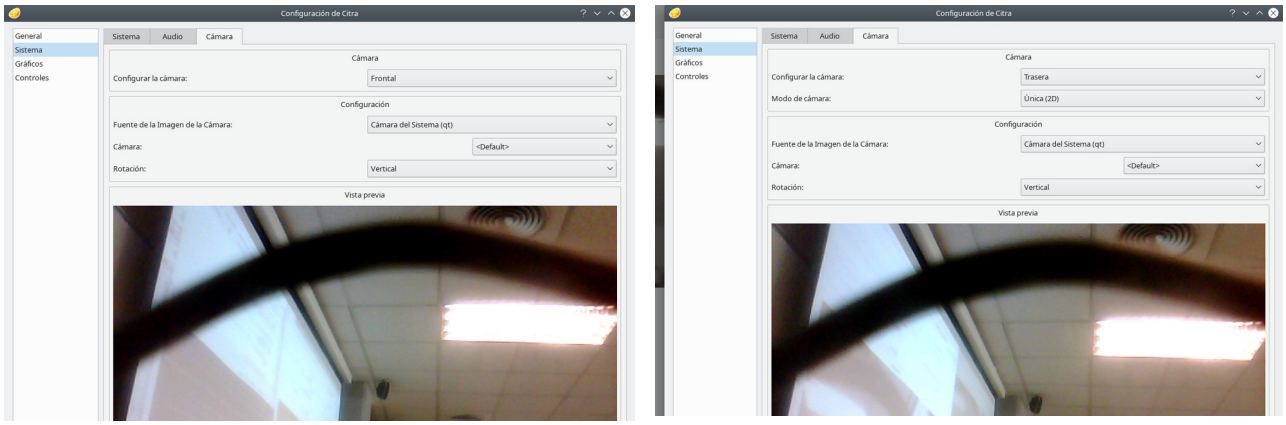


Figura 5: Detalle de la parte trasera de la consola donde se observan las dos cámaras: Outer cameras (imagen obtenida de <https://static.giantbomb.com/uploads/scale_super/8/82063/2686372-3ds_back.jpg>).

Para poder ejecutar estos ejemplos en *Citra*, **utilizaremos la versión c*itra-qt***, antes será necesario configurar el uso de la cámara mediante el menú *Emulation* | *Configure* .... Este abre una caja de diálogo, como la que muestra la Figura 6, en la que el apartado *Sistema* contiene la pestaña *Cámara* que permite escoger cuál de las cámaras de la consola (la frontal o las traseras) va a ser emulada con:

- Una de las cámaras conectada al sistema.
- Un fichero de los que estén disponibles en nuestro sistema de archivos (y, por tanto, siempre obtendremos esa imagen fija). Esta es la opción que usaremos en caso de no disponer de una cámara activa en nuestro sistema .
- O “Vacío (nada) “, es decir, que no está operativa (devolviendo siempre una imagen en negro).



a) b)

Figura 6: Caja de diálogo para configuración de la emulación de las cámaras que ofrece "Citra" para la cámara frontal (a) y las traseras (b).

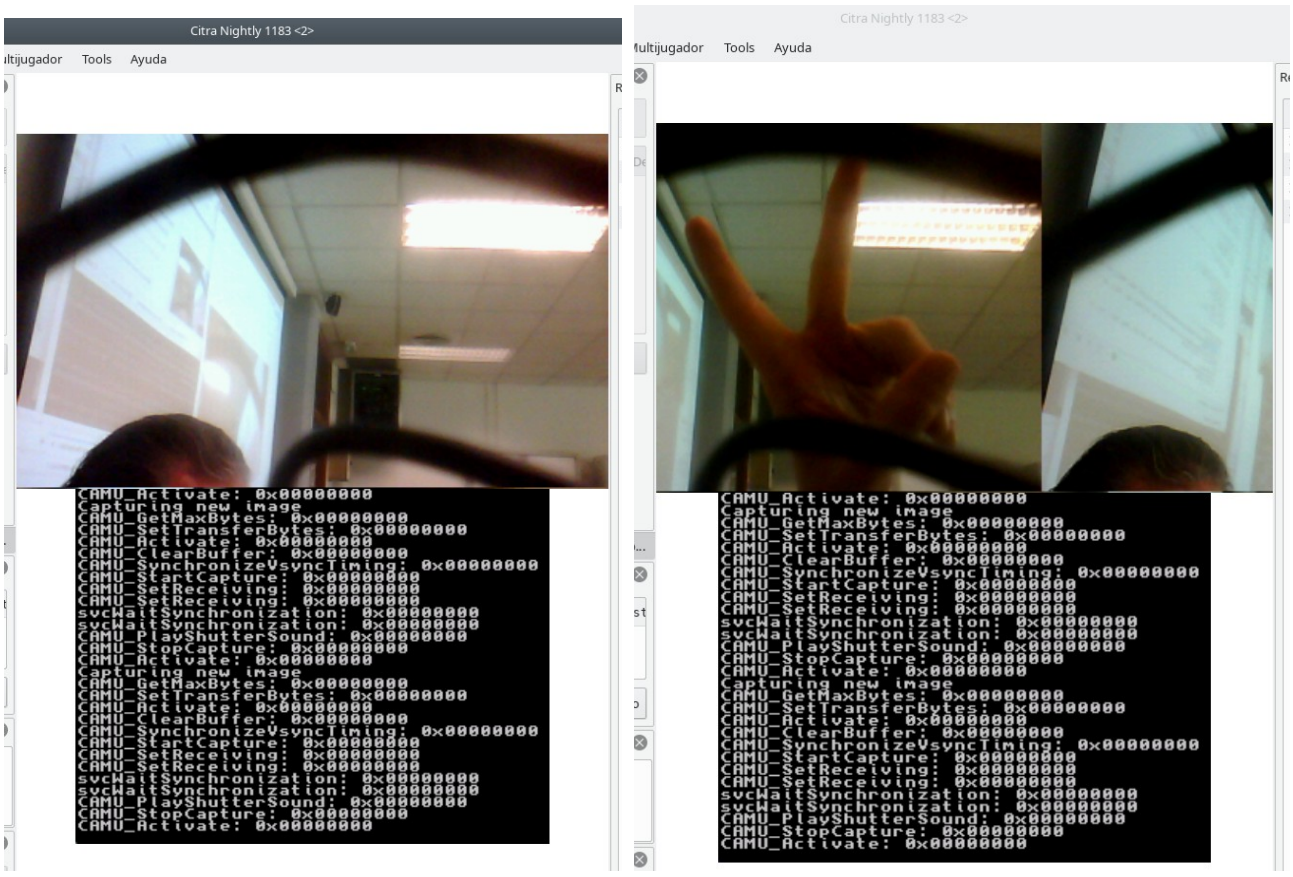


Figura 7: Dos instantes de captura de la imagen de la cámara.

La Figura 6a muestra cómo se ha seleccionado la cámara frontal para ser emulada con una cámara web que utilizamos en el laboratorio. Así, en función de lo que hayamos configurado, será posible ejecutar el ejemplo mencionado. En nuestro caso hemos de seleccionar (como se muestra en la Figura 6b) en el bloque de "Cámara", dentro de "Configurar la cámara" con el valor de "Trasera" y en "Modo de cámara" seleccionaremos el valor "Única (2D)"[9] y, ya dentro del apartado

9 Podríamos seleccionar en "Modo de cámara" el valor "Doble (3D)" y, entonces aparecerá el "Posición de cámara"

"Configuración", podremos escoger como "Fuente de la imagen de la Cámara" el valor de "Cámara del Sistema (Qt)", lo que nos permitirá escoger una de las que tengamos conectadas bajo el desplegable de "Cámara". Utilizando el botón del apartado de "Vista previa" podremos ver en directo lo que la cámara esté capturando.

Ejercicio 8. Examine el código del ejemplo *3ds-examples-master/camera/image* para observar, en el código de la función *takePicture3D,* qué secuencia de pasos realiza y que se utilizan las dos cámaras traseras (SELECT_OUT1_OUT2) para esta acción. Anote la secuencia de llamadas a los servicios de cámara (empiezan con el prefijo **CAMU_**).

Por otro lado, para poder tomar una secuencia de vídeo veamos el ejemplo *3ds-examples-master/camera/video* que es el que describe cómo se captura una serie de cuadros a través de la librería *libctru*. Este ejemplo permite obtener un vídeo, véase Figura 8, que se muestra un momento de la visualización de la captura junto a la configuración de la cámara empleada.

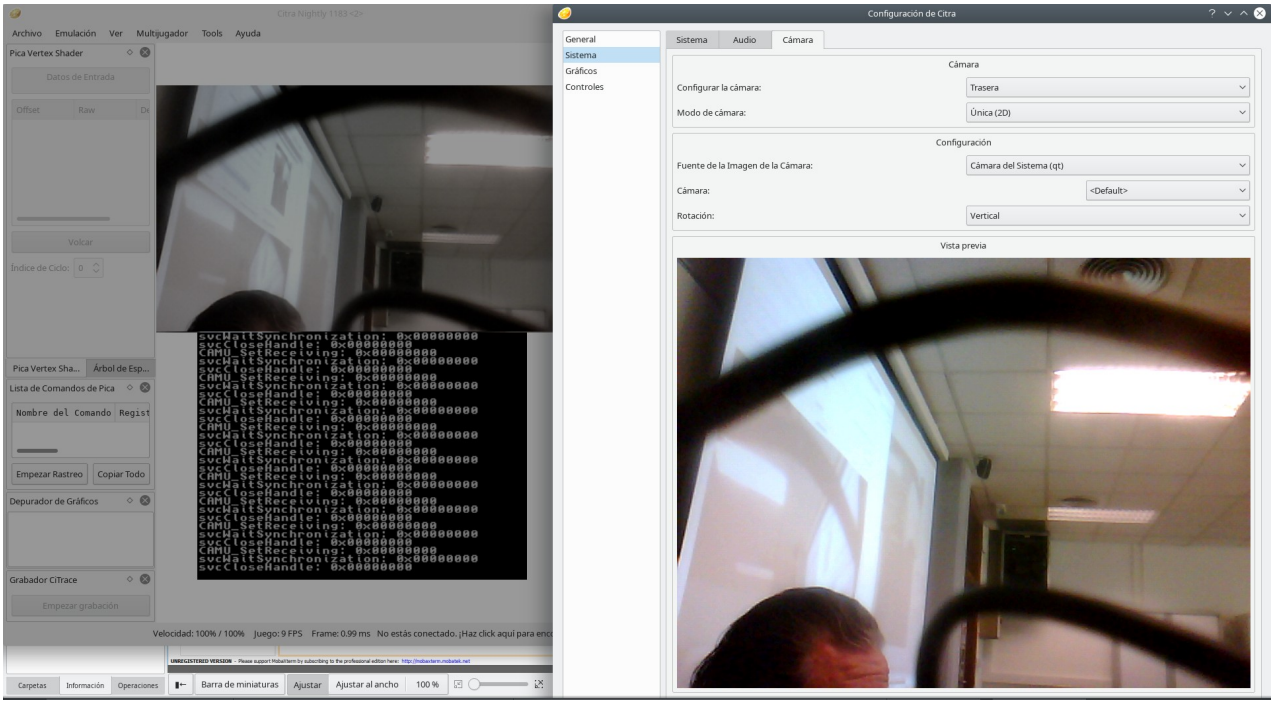


Figura 8: Captura de vídeo utilizando la configuración de cámara trasera única.

Ejercicio 9. Examine el código del ejemplo *3ds-examples-master/camera/video* para observar, en el código de la función *main,* qué secuencia de pasos realiza y que se utilizan las dos cámaras traseras (SELECT_OUT1_OUT2) para esta acción. Anote la secuencia de llamadas a los servicios de cámara (empiezan con el prefijo **CAMU_**).

las opciones de "Derecha" e "Izquierda", lo que permitiría seleccionar una cámara física diferente para emular las dos cámaras traseras de la *Nintendo 3DS* y ofrecer una visión estereoscópica. Como el emulador *Citra*, a fecha de hoy, no soportar este modo de visión, no vamos a realizar esta prueba; pero está en la lista de modificaciones que se están realizando ("Add Support for Stereoscopic 3D #3632" <https://github.com/citra-emu/citra/pull/3632>), así que igual pronto … En el curso 2k20/2k21, ya debería estar disponible este modo.

4 Trabajo autónomo

Junto a este boletín hay un fichero *mostresMisterioses.c* que guarda un *buffer* de 8.178 elementos, de tipo *byte* (*u8*), que han sido adquiridos a una frecuencia de muestreo de 11025 Hz y empleando un canal (sonido monofónico). Describa con sus propias palabras qué contenido esconde esta señal de audio, para ello genere una versión del ejemplo de *audio/streaming del apartado 2.1 Síntesis por generación de tonos*, con una versión de la función "fill_buffer" adaptada para cargar y reproducir esta señal.

También es momento de guardar un resultado de la ejecución de los ejemplos del apartado 3 Cámara, para el porfolio. Para ello habrá de obtener una captura del escritorio donde se vea su imagen obtenida ejecutando el ejemplo de la cámara y el de captura de vídeo, al estilo de la Figura 9. Anote la secuencia de pasos que diferencian el ejemplo de toma de fotografías (instantáneas) del de captura de vídeo vistos en el apartado 3 Cámara. ¿Qué diferencias existen entre la secuencia de pasos del ejemplo de la cámara con el ejemplo de captura de vídeo?

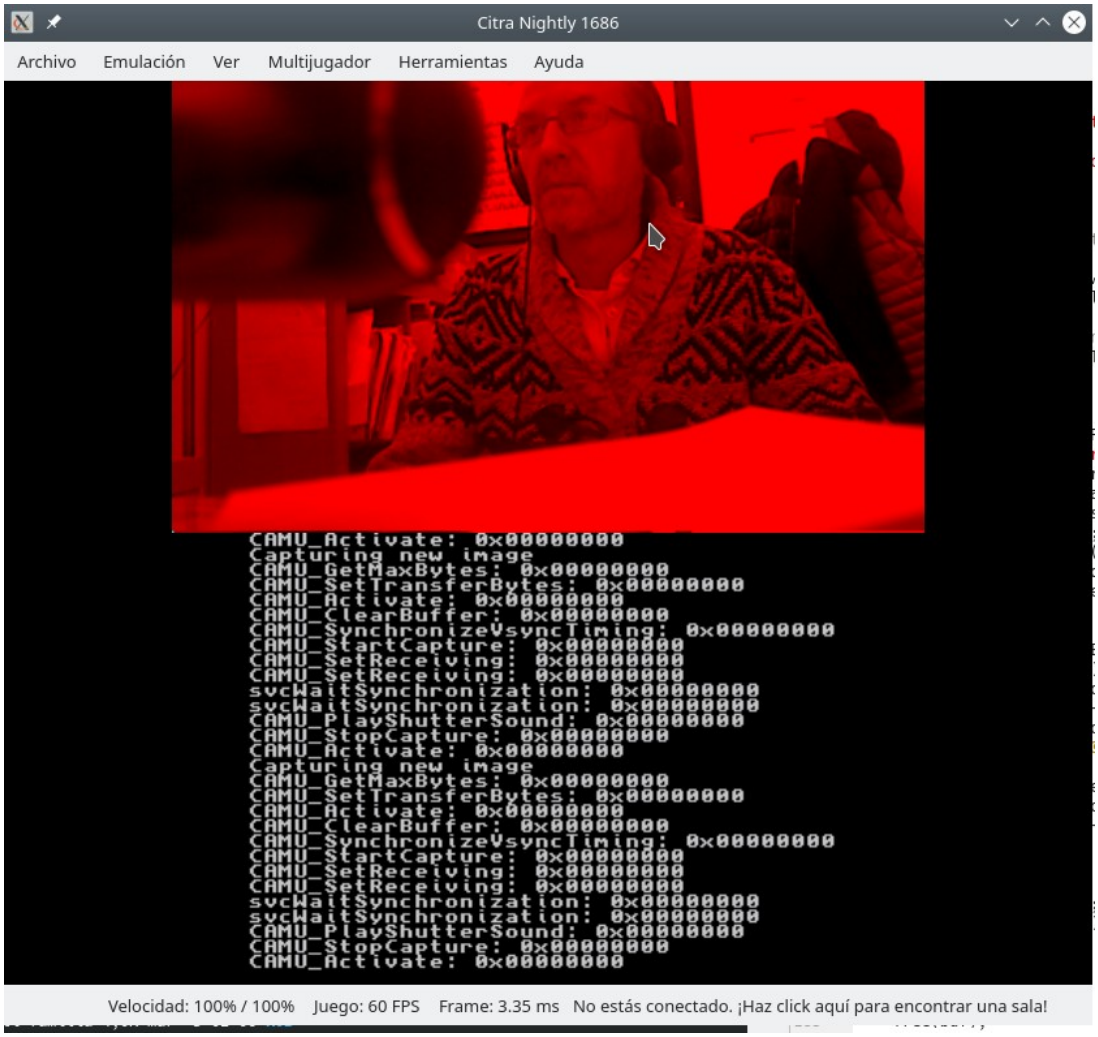


Figura 9: Captura de pantalla del ejemplo de uso de la cámara en Citra.

5 Bibliografía

[1] *Services API*. Disponible en la URL <https://www.3dbrew.org/wiki/Services_API>.

[2] “Smealum”. “libctru - CTR User Library”. Disponible en la URL <http://smealum.github.io/ctrulib/>.

[3] *HLE Audio Comes to Citra*. Disponible en <https://citra-emu.org/entry/hle-audio-comes-to-citra/>.

Anexo I – Configuración de Citra

En este apartado haremos referencia a las modificaciones que hay que hacer en la instalación de *Citra* para que habilite la reproducción de sonido. Para poder hacer uso del sonido en la 3DS se utiliza el DSP que incorpora Este procesador carga un fichero que contiene un componente binario que recibe el nombre de *dspfirmware.cdc* y que le permite decodificar el audio en formato WAVE, así como también algunos formatos de audio comprimido (MP3, AC3 o ACC).

Para emular este componente hay que crear un fichero vacío en la instalación de *Citra* y esto suponen pequeñas diferencias en un sistema operativo u otro. Veamos cómo llevarlo a cabo en un sistema Linux (que debería ser aplicable a otros Unix como macOS) y en Windows 10. Si dispones de la consola real, podrías descargar el fichero de la misma[10] con utilidades como *DSP1* o *DSP Dump*[11].

En nuestro caso, en el laboratorio, asumiremos que no disponemos del equipo físico para permitir que el mayor número de personas puedas realizar esta práctica. Así que hemos de simular con un fichero vacío la existencia de ese componente para que la emulación de *Citra* del sonido siga adelante y lo active. Para ello, dependiendo del Sistema Operativo donde se ejecuta *Citra*, habrá que crear el directorio correspondiente dentro del que sirve para emular la tarjeta de memoria y el fichero que ya hemos mencionado.

En el caso de GNU/Linux esto se traduce en ejecutar en un terminal:

```
$ mkdir  ~/.local/share/citra-emu/sdmc/3ds
$ echo >  ~/.local/share/citra-emu/sdmc/3ds/dspfirm.cdc
```

y como resultado veremos la situación que nos muestra la Figura 10.

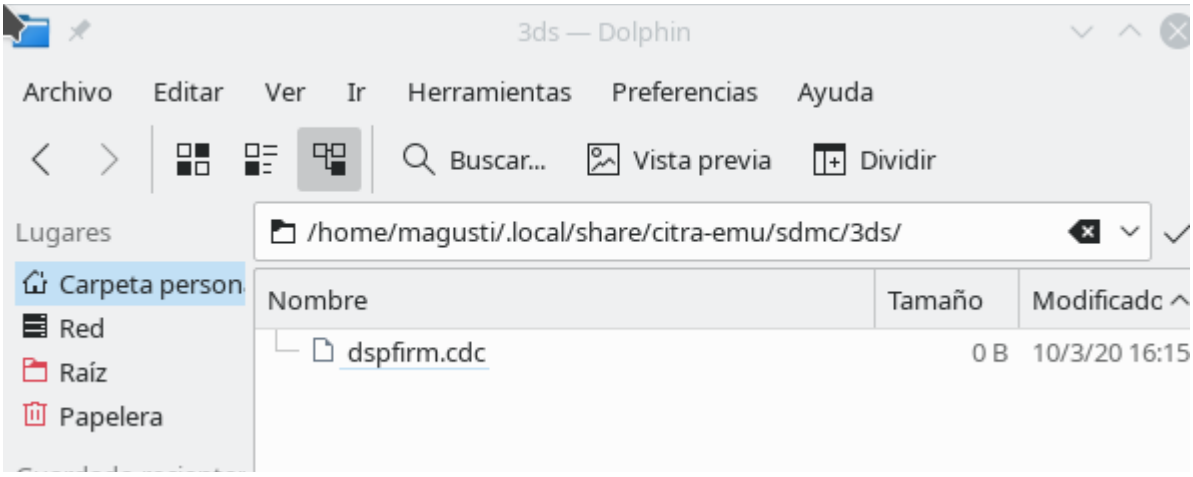


Figura 10: Fichero "dspfirm.cdc" vacío creado en la cuenta de un usuario en Linux.

En el caso de Windows 10, el directorio es el que muestra la Figura 11 y el fichero se puede crear también desde el terminal o desde el propio administrador de archivos.

10 En "Can't play MP3 files #44" <https://github.com/deltabeard/ctrmus/issues/44> se eplica cómo hacerlo.

11 Se pueden encontrar estas utilidades en el GitJub de DSP1<https://github.com/zoogie/DSP1/releases> o en el de DSP Dump <https://github.com/Cruel/DspDump>.

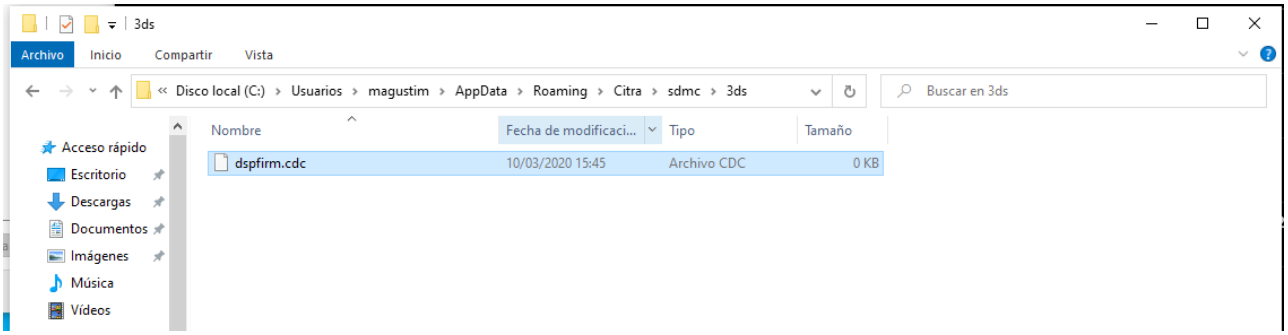


Figura 11: Fichero "dspfirm.cdc" vacío creado en la cuenta de un usuario en Windows 10.

Ahora sí que funcionan los ejemplos del SDK como *audio/streaming* o *audio/filters*, como se muestra en la Figura 12.

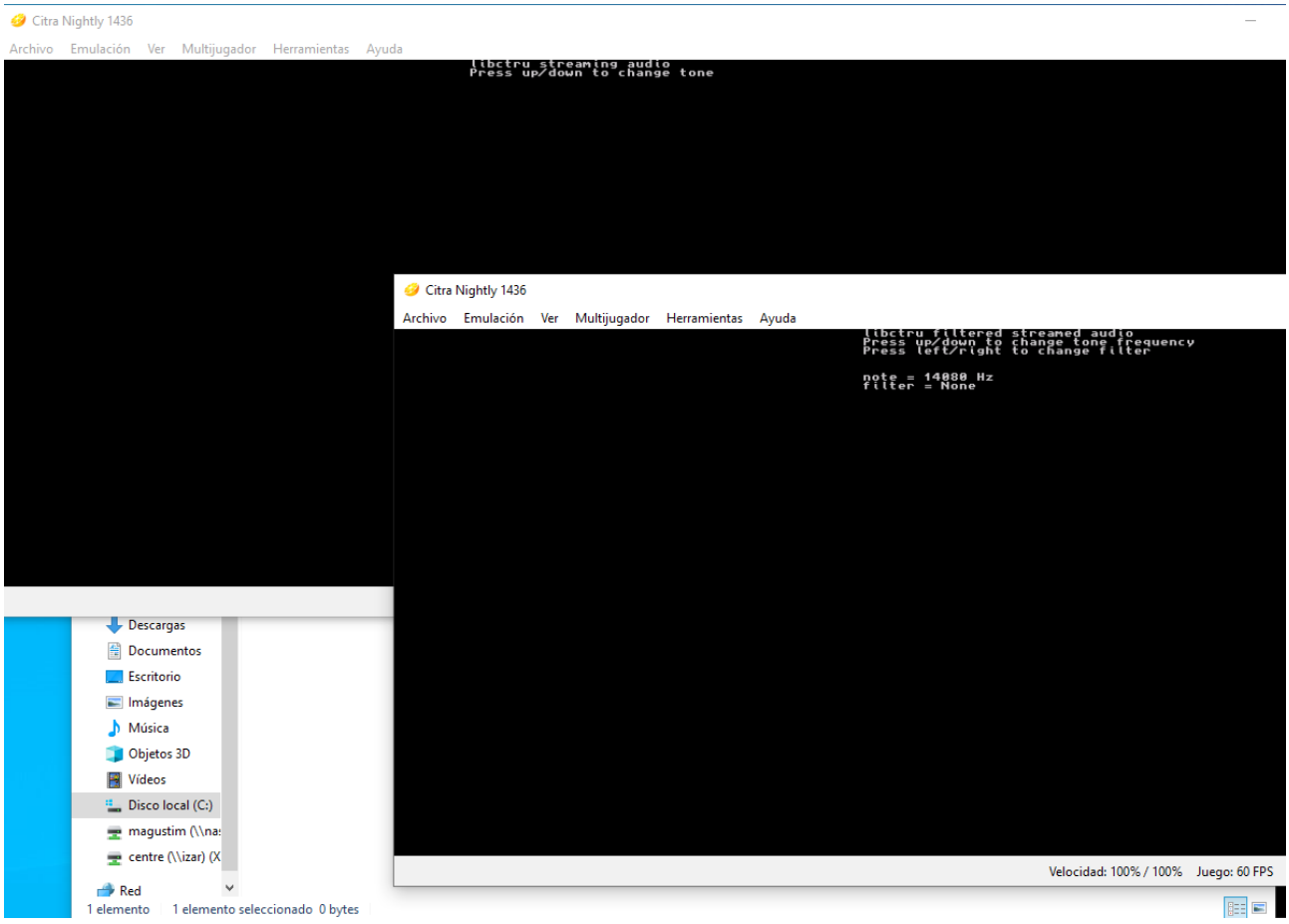


Figura 12: Salida de los ejemplos de audio del SDK "streaming" y "filters".

También se puede probar con un reproductor de formatos de audio como "3DS Music Player"[12], para el que se probado con éxito con diferentes formatos de ficheros de audio, para lo que se ha procedido a copiar en *~/.local/share/citra-emu/sdmc/exemplesAudio* archivos de formato:

- WAVE (obtenido de los de sistema disponibles en el sistema), mediante la creación de enlaces simbólicos.

12 Disponible en la URL <https://github.com/deltabeard/ctrmus>.

- OPUS de los ejemplos de *Switch* en *$DEVKITPRO/examples/switch/audio/hwopus-decoder/data/sample.opus*.
- OGG de los ejemplos en */usr/share/sound* o de los de *GameCube* en *$DEVKITPRO/examples/gamecube/audio/oggplayer/data/sample.ogg*.

Así que nuestras aplicaciones podrán hacer uso del sonido, reproduciendo los archivos cuyo formato sea conocido, como p. ej. los RAW o aquellos para los que existen librerías que puedan decodificarlos y, por tanto, nos resuelvan extraer el conjunto de muestras de sonido que contienen en formato plano (ADPCM, PCM o RAW).