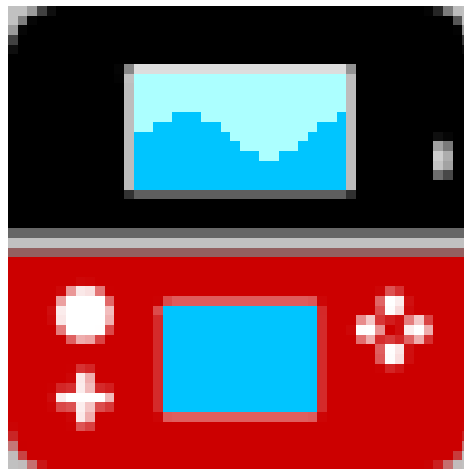


# Arquitectura y Entornos de desarrollo para videoconsolas

## **Práctica 2**

### **Acceso a información en el hardware de la videoconsola 3DS**



M. Agustí  
Febrero 2019

La práctica se va a realizar en entorno GNU/Linux en el laboratorio. Vamos a abordar el acceso a la información básica de la consola: como personalización, gestión del tiempo y gestión de la entrada de datos del usuario.

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

## 1 Introducción

Trabajaremos a partir de unos ejemplos de código disponibles en la documentación del SDK de *devKitPro*<sup>1</sup> y con la documentación disponible [1]. Tenga pues a mano los ejemplos que en la anterior práctica se copiaron en el espacio del usuario. Recuerde que esto era necesario para poder compilarlos, pero que puede llevarse el directorio base del proyecto a cualquier directorio de su espacio de usuario. Las rutas a los ejemplos 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. Para tener disponible el contenido de la documentación de libctru en local, en el Anexo I Documentación en local de libctru, citro3d y citro2d encontrará las instrucciones para obtener un PDF a partir del código fuente de la librería.

Veamos que hay una buena parte del API de NDS que se encuentra en las siguientes plataforma en la escala de desarrollos en el tiempo que ha hecho la compañía que diseñó estas plataformas, en este caso la 3DS. Se trata del manejo del terminal o la consola, esto es la utilización básica del texto en pantalla; que, por cierto, es equivalente al de los terminales en las plataformas de escritorio. Después pasaremos a ver cuestiones más dependientes de la plataforma como la gestión de información y la del tiempo. Veremos en esta práctica cómo acceder a informaciones muy básicas como parámetros de configuración del dispositivo y gestión del tiempo, así como a las operaciones de entrada y salida con el hardware del dispositivo y con los sistemas de archivos..

## 2 Cuestiones básicas

Revisemos primero cual es el ciclo de desarrollo de las aplicaciones bajo DevkitPro para la plataforma de 3DS. El proyecto básico para desarrollo en la 3DS se estructura en torno a una disposición de directorios similar a lo visto para el caso de NDS. La Figura 1 muestra el ejemplo de 3DS application (en *templates/application/*) que constituye una plantilla para empezar con los proyectos sobre 3DS con *libctru*.

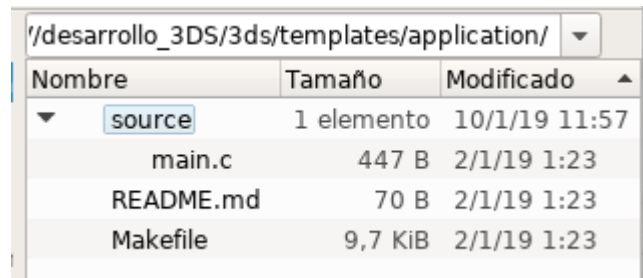
Este proyecto es equivalente al que se puede encontrar en *graphics/printing/hello-world*. En ambos casos y, solo con la salvedad de un ligero cambio de orden en la secuencia de instrucciones, ambos ejemplos de basan en el algoritmo siguiente.

- Inicializa la consola en la pantalla superior → *consoleInit*
- Bucle principal de gestión de eventos → *while (aptMainLoop())*

---

<sup>1</sup>En Sourceforge: <<http://sourceforge.net/projects/devkitpro/files/libnds/>> y la documentación está en línea en “libctru - CTR User Library” <<https://libctru.devkitpro.org/>>.

- Actualización de la salida a las pantallas → `gfxSwapBuffers` + `gspWaitForVBlank`
- Gestión de los eventos de la botonera → `hidScanInput` + `hidKeysDown`
- Liberación de recursos → `gfxExit`



| Nombre    | Tamaño     | Modificado    |
|-----------|------------|---------------|
| source    | 1 elemento | 10/1/19 11:57 |
| main.c    | 447 B      | 2/1/19 1:23   |
| README.md | 70 B       | 2/1/19 1:23   |
| Makefile  | 9,7 KiB    | 2/1/19 1:23   |

a) Contenido del directorio.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <3ds.h>

int main(int argc, char* argv[])
{
    > gfxInitDefault();
    > consoleInit(GFX_TOP, NULL);

    > printf("Hello, world!\n");

    > // Main loop
    > while (aptMainLoop())
    > {
        > > gspWaitForVBlank();
        > > gfxSwapBuffers();
        > > hidScanInput();

        > > // Your code goes here
        > > u32 kDown = hidKeysDown();
        > > if (kDown & KEY_START)
        > > > break; // break in order to return to hbmenu
    > }

    > gfxExit();
    > return 0;
}
```

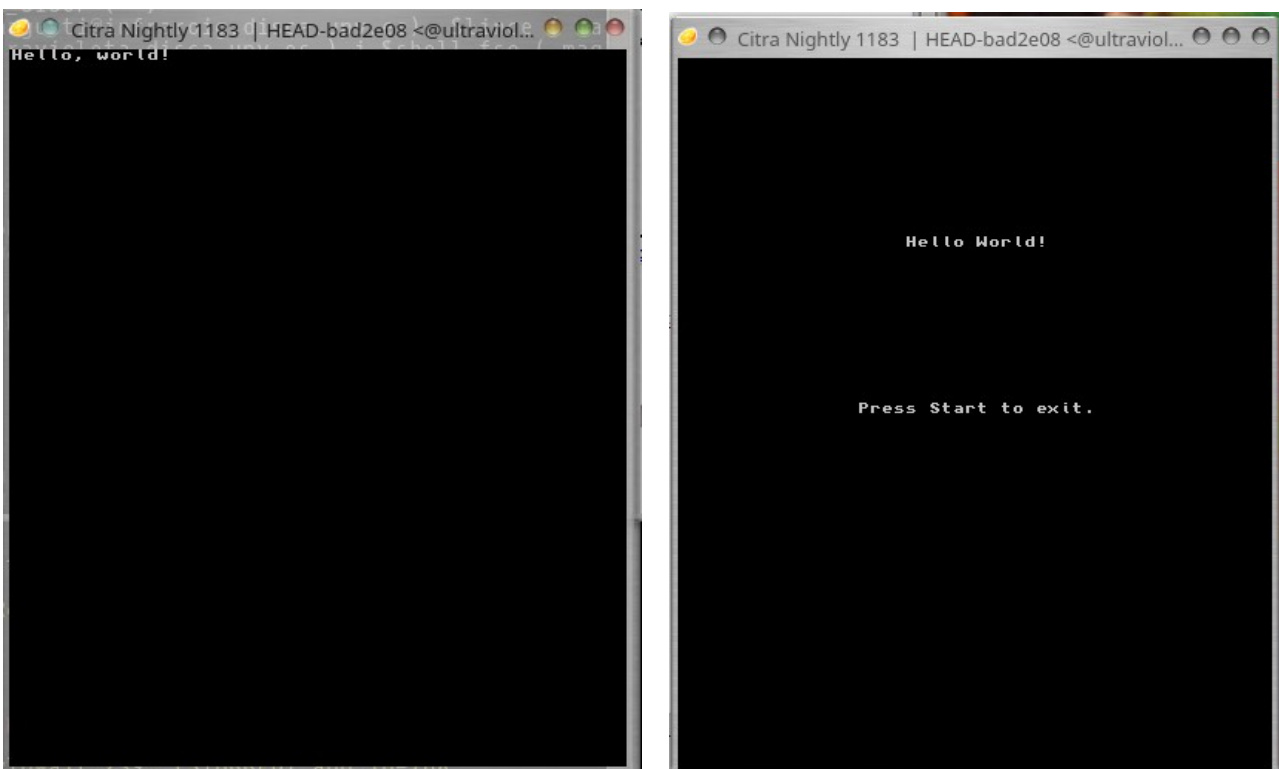
b) Contenido del código fuente.

Figura 1: Estructura del proyecto básico para desarrollo sobre 3DS.

Para generar el ejecutable o distribuible para la 3DS haremos uso de la orden *make* (como venimos haciendo en el caso de NDS) dentro del directorio base del proyecto en curso y cuyo nombre será asignado al resultado obtenido.

Para ello el fichero *Makefile* incluye una serie de variables que configura su proceso y entre las que destacaremos:

```
#-----  
# TARGET is the name of the output  
# SOURCES is a list of directories containing source code  
# DATA is a list of directories containing data files  
# INCLUDES is a list of directories containing header files  
# GRAPHICS is a list of directories containing graphics files  
# GFXBUILD is the directory where converted graphics files will be placed  
...  
# ROMFS is the directory which contains the RomFS, relative to the Makefile (Optional)  
# APP_TITLE is the name of the app stored in the SMDH file (Optional)  
# APP_DESCRIPTION is the description of the app stored in the SMDH file (Optional)  
# APP_AUTHOR is the author of the app stored in the SMDH file (Optional)  
# ICON is the filename of the icon (.png), relative to the project folder.  
...
```



*Figura 2: Resultado de la ejecución del ejemplo (a) `template/application` y (b) `graphics/printing/hello-world`*

El resultado obtenido ya es posible ejecutarlo con la ayuda de un emulador en la propia máquina de desarrollo, como p. ej. Citra. Compruebe que la salida es, tanto para este ejemplo *application* (*template/application*) como para el de *hello-world* (*graphics/printing/hello-world*) la que se muestra en la Figura 2.

**Ejercicio 1.** Busque en la documentación el cometido de las funciones *consoleInit*, *aptMainLoop*, *gfxSwapBuffers*, *gspWaitForVBlank*, *HidScanInput*, *hidKeysDown* y *gfxExit*.

**Ejercicio 2.** Se ha comentado en la explicación que las dos aplicaciones son equivalentes, pero la salida de las mismas (véase la Figura 2) es diferente. Observe el código fuente de ambos ejemplos ¿Qué hay de diferente en las dos aplicaciones para que la salida no sea idéntica?

La orden *make* genera 3 ficheros que se corresponden con los formatos:

- *smdh*. Que es el formato de un archivo<sup>2</sup> que contendrá el icono y la descripción textual de la aplicación que construimos. Puede ver el contenido utilizando la aplicación web *SMDH Creator*<sup>3</sup> y que se muestra en la Figura 3. Si no les hemos asignado nada, el SDK nos proporciona estos valores por defecto, que incluye un PNG de resolución 48x48, 8-bits y non entrelazado

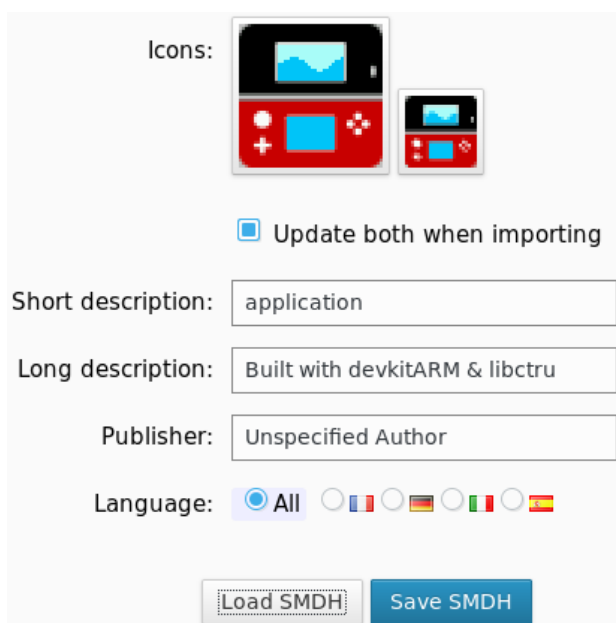


Figura 3: Uso de la aplicación web *SMDH Creator* para ver el contenido del fichero *SMDH* del ejemplo.

- *3dsx*. El fichero en formato<sup>4</sup> ejecutable que podemos utilizar con el emulador, p. ej. Citra.
- *elf*. Que es un formato binario con la información de los símbolos para poder depurar el proyecto.

### 3 Operaciones básicas de texto sobre pantalla

Nos referimos en este apartado a cuestiones que tienen que ver con el uso de las pantallas en modo

<sup>2</sup> Más detalles sobre este formato en la URL <<https://www.3dbrew.org/wiki/SMDH>>.

<sup>3</sup> Disponible en la URL <[http://usuarios.tinet.cat/mark/smdh\\_creator/](http://usuarios.tinet.cat/mark/smdh_creator/)>.

<sup>4</sup> Más información al respecto de este formato en la URL <[https://www.3dbrew.org/wiki/3DSX\\_Format](https://www.3dbrew.org/wiki/3DSX_Format)>.

consola o terminal. Esto es, el uso de texto para mostrar información al estilo de como se hace en las aplicaciones de escritorio.

Empecemos por ver cómo poner texto en las dos pantallas, por cierto de resolución de texto de 30 filas por 50 columnas para la pantalla superior y de 30x40 para la inferior. Véase el resultado de ejecutar el ejemplo de 3DS *graphics/printing/both-screen-text* como lo muestra la Figura 4.

**Ejercicio 3.** Compare el ejemplo de 3DS *graphics/printing/both-screen-text* con el de NDS *print\_both\_screens* (en *Graphics/Printing/print\_both\_screens*) y observe que se utiliza el mismo nombre para el tipo de las pantallas de ambas plataformas *PrintConsole*, así como las funciones *consoleInit* y *consoleSelect*.

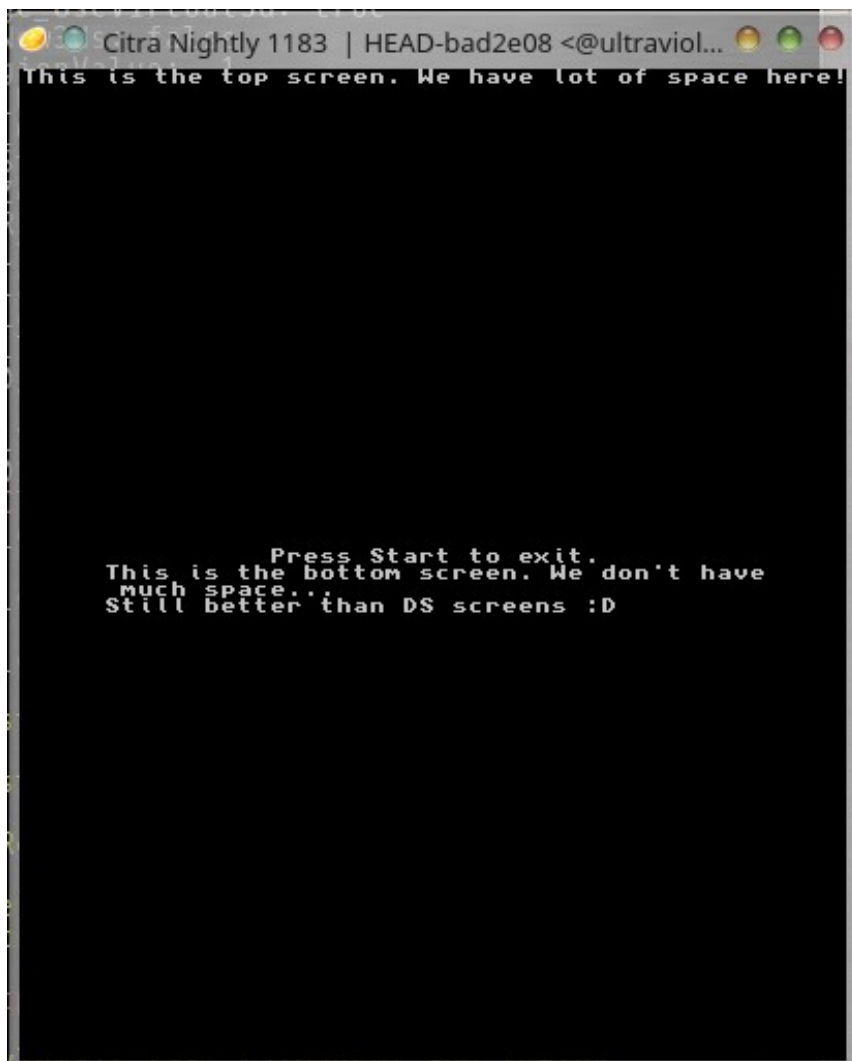


Figura 4: Resultado de la ejecución de *.graphics/printing/both-screen-text*.

Usando las secuencias de escape [3] ya vistas en el ejemplo de 3DS *hello-world* (*graphics/printing/hello-world*) es posible posicionar el texto y darle estilo (color al texto o al fondo del texto) como muestra la Figura 5. El mecanismo se puede utilizar en las aplicaciones de escritorio que tienen salida en la consolas de texto y también está presente en la plataforma NDS (véase el ejemplo de *Graphics/Printing/ansi\_console*).

Véase el resultado de ejecutar el ejemplo de 3DS *graphics/printing/colored-text* como lo muestra la

Figura 5.

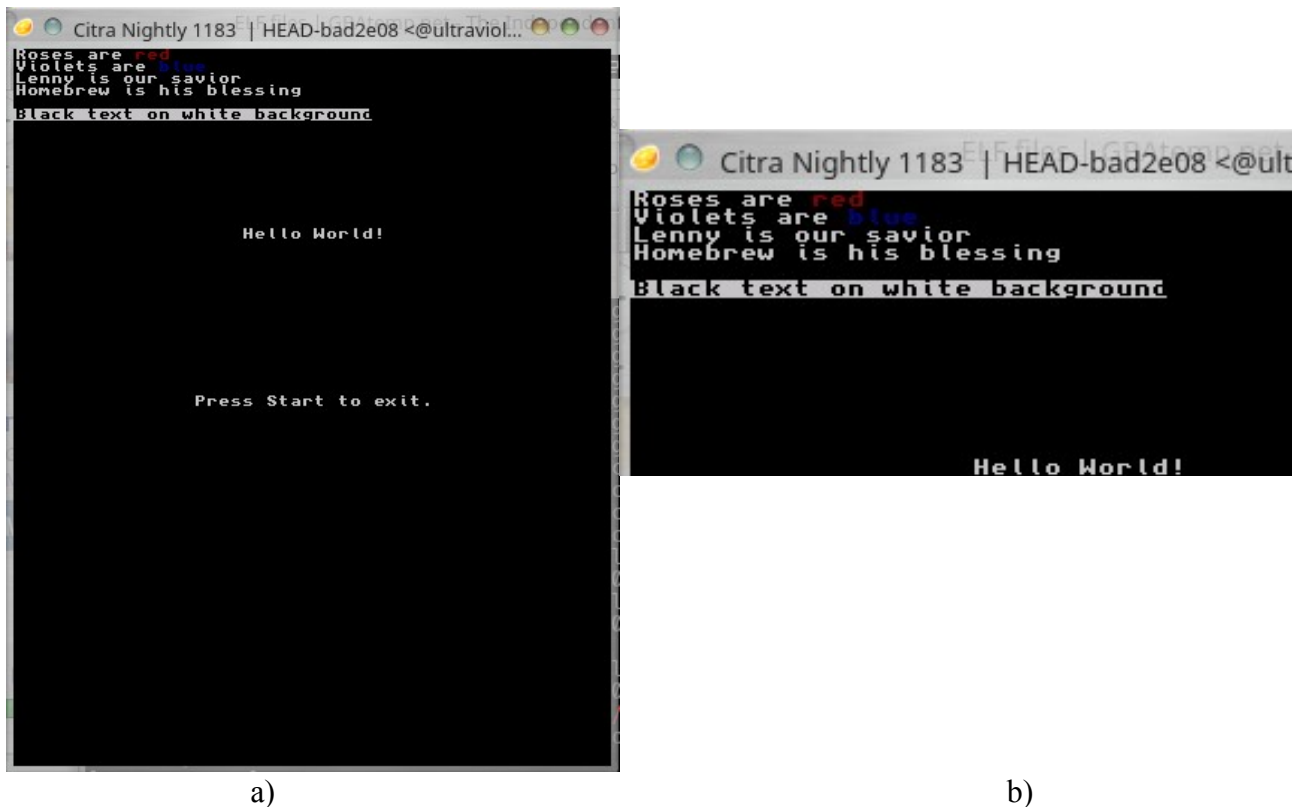


Figura 5: Resultado de la ejecución de colored-text (a) y detalle ampliado de la pantalla superior (b).

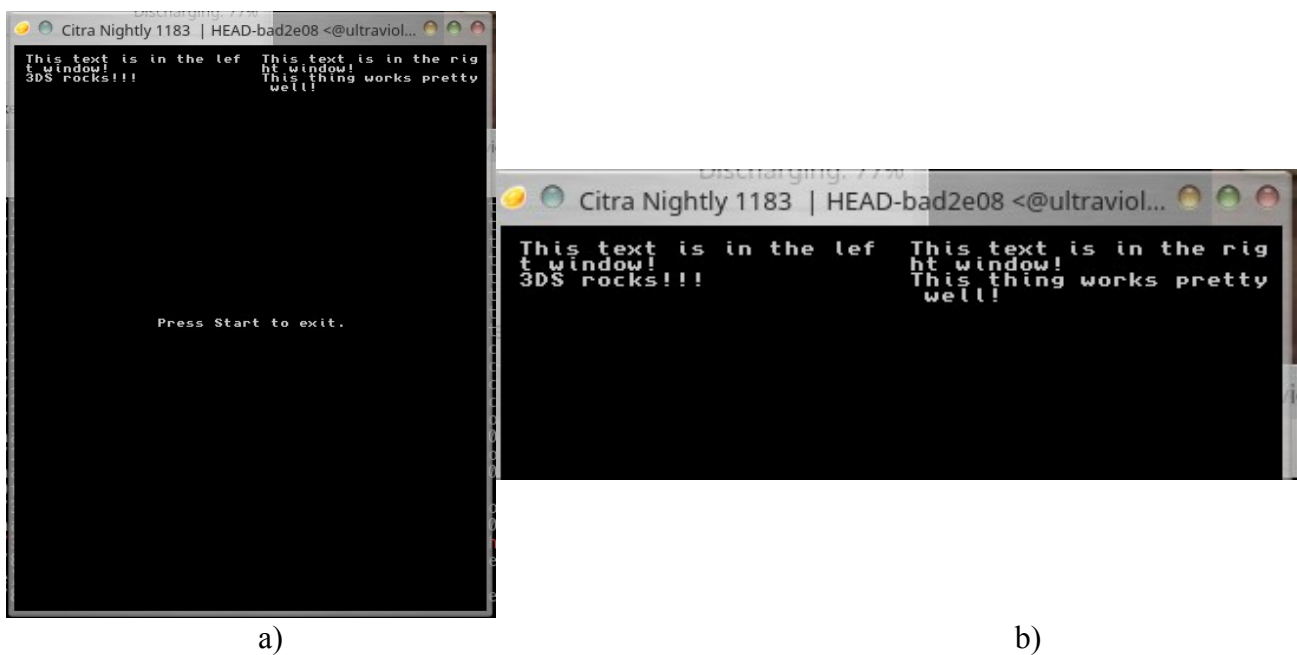


Figura 6: Salida de multiple-windows-text (a) y detalle ampliado de la pantalla superior (b).

**Ejercicio 4.** Anote las secuencias de escape que se pueden observar en los ejemplos de 3DS *hello-world* (*graphics/printing/hello-world*) y *coloredtext* (*graphics/printing/colored-text*), indicando

cómo se especifican las coordenadas donde se imprimirá el texto y cómo se indica el color del texto (primer plano) y el de fondo .

Es posible declarar “ventanas” o espacios para el texto, de forma que es el propio sistema el que organiza la disposición del texto que generamos de acuerdo con la definición del espacio asignado a estas áreas. Véase el resultado de ejecutar el ejemplo de 3DS *graphics/printing/multiple-windows-text* como lo muestra la Figura 6.

**Ejercicio 5.** Compare el ejemplo de 3DS *graphics/printing/multiple-windows-text* con el de NDS *Graphics/Printing/console\_windows* y observe que se utiliza el mismo nombre para el tipo de las pantallas de ambas plataformas *PrintConsole*, así como las funciones *consoleSetWindow* y *consoleSelect*. En el ejemplo de NDS se pone borde a las “ventanas” ¿Se puede trasladar aquí esa “decoración”?

## 4 Acceso al sistema

En este apartado revisamos cómo acceder a información del sistema operativo y del hardware de la consola. Esto nos llevará a ver cuestiones de configuración y la gestión del tiempo.

Empecemos por ver la gestión de información relativa al sistema. Para ello el ejemplo de 3DS *get\_system\_language* nos muestra el resultado de la Figura 7a.. Observe que Citra dispone en el menú *Emulación | Configurar ...* con una caja de diálogo que permite especificar el valor del idioma, junto con otros parámetros. En este caso el valor de “Language code” devuelto (5) corresponde con el del idioma español.

Esta correspondencia se puede comprobar revisando el contenido del servicio de configuración o *CFGU (Configuration) Service* que está implementado en el fichero *cfgu.h*<sup>5</sup>. En él se verá la declaración de *CFG\_Language* y cómo aparece el idioma español (*CFG\_LANGUAGE\_ES*) asignado con el valor 5. En el mismo fichero también podrá encontrar, , entre otras, información relativa a las regiones (*CFG\_Region*) o el modelo de la consola (*CFGU\_GetSystemModel* y *CFGU\_GetModelNintendo2DS*)

**Ejercicio 6.** Modifique el ejemplo de 3DS *get\_system\_language* para que muestre una cadena de texto que identifique el idioma en el que se encuentra configurada la consola y cambie la configuración en el emulador (Figura 7b), comprobando si se refleja en la ejecución del programa.

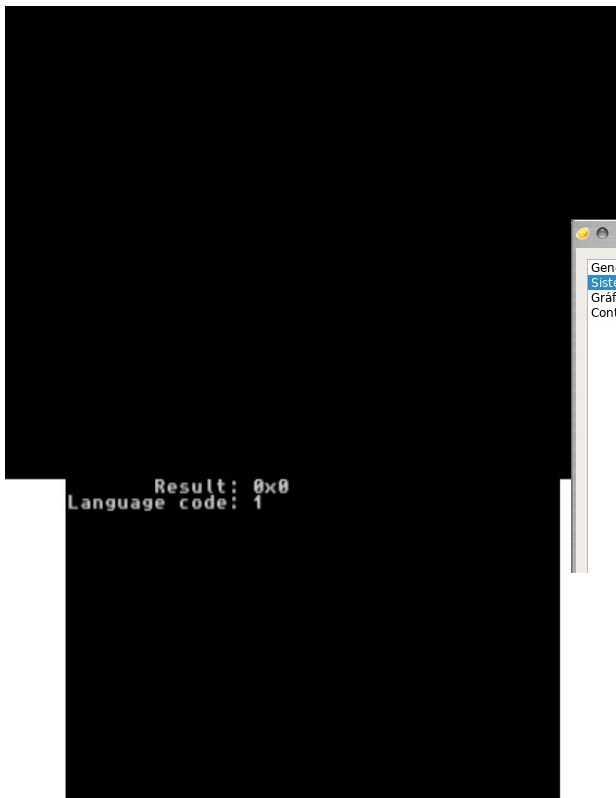
Relacionado con los parámetros del sistema están también los del sistema operativo que gestiona la consola, denominado *OS related stuff* en la documentación. En este sentido, el fichero que lo implementa es *os.h*<sup>6</sup> . En él se encontrarán cosas como el estado del control de visualización estereoscópica de la pantalla superior (*osGet3DSliderState*), la versión del *firmware* del equipo (*osGetFirmVersion* ) o la potencia de la señal WiFi recibida (*osGetWifiStrength*).y que se corresponde con cuatro niveles discretos y que se muestran en el menú *Home* de la consola.

Una cuestión que pertenece a este grupo de informaciones que nos ofrece el sistema es el paso del tiempo. En este sentido tenemos la posibilidad de obtener el tiempo con la precisión de milisegundos (con *osGetTime* ) y utilizar temporizadores (*osTickCounterStart*, *osTickCounterUpdate* y *osTickCounterRead*).

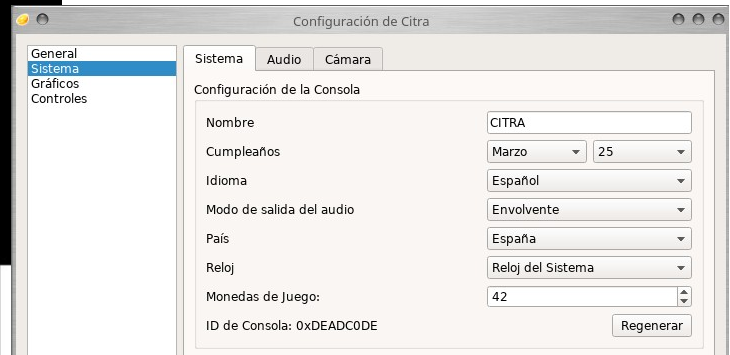
5 Este fichero está instalado en la máquina en *\$DEVKITPRO/libctru/include/3ds/services/cfgu.h* y también en la URL [http://smealum.github.io/ctrulib/cfgu\\_8h.html](http://smealum.github.io/ctrulib/cfgu_8h.html).

6 Este fichero está instalado en la máquina en *\$DEVKITPRO/libctru/include/3ds/os.h* y también en la URL [http://smealum.github.io/ctrulib/os\\_8h.html](http://smealum.github.io/ctrulib/os_8h.html)



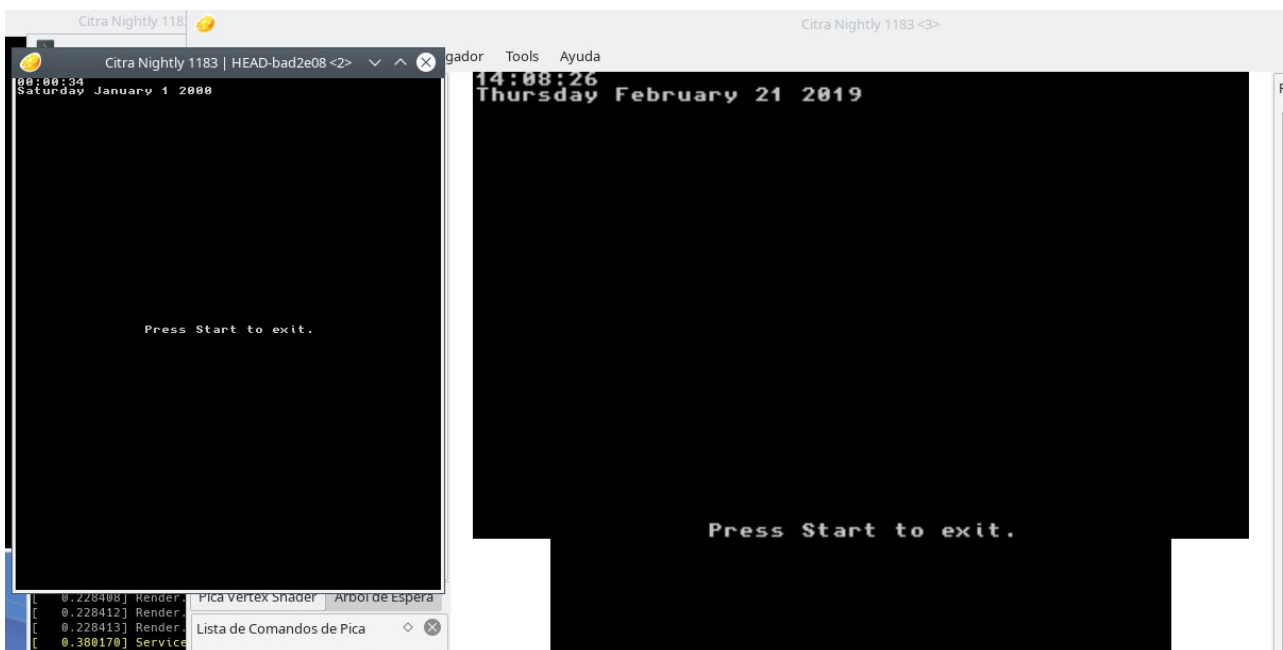


a)

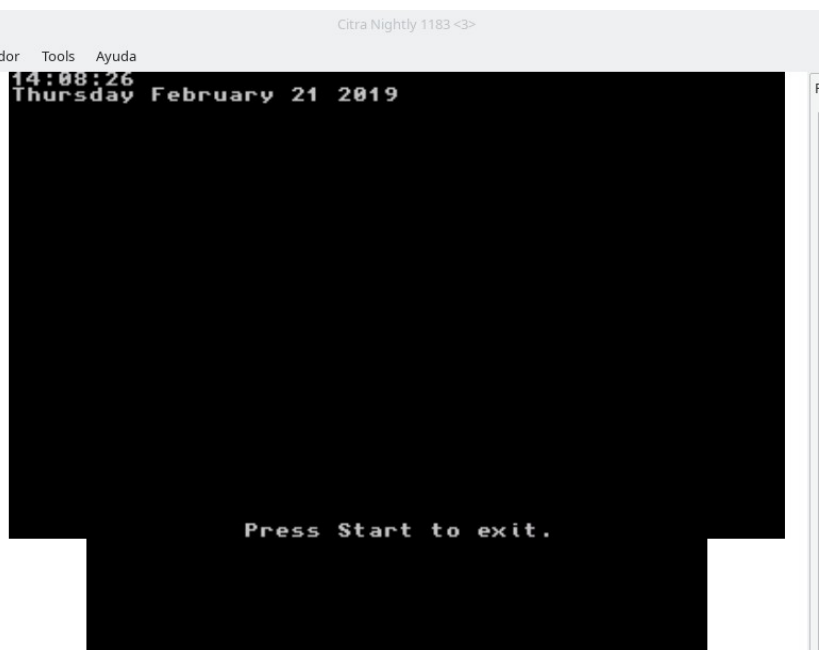


b)

Figura 7: Salida de `get_system_language` (a) y configuración de Citra-qt (b)..



a)



b)

Figura 8: Resultado de la ejecución del ejemplo `rtc` de 3DS en (a) citra y (b) citra-qt..

En este momento vamos a ver un ejemplo de obtención de tiempo con algo menos de precisión,

pero que sigue siendo de utilidad en un buen número de aplicaciones. Se trata del ejemplo *rtc* (en *time/rtc*) que se muestra en la Figura 8.

Este mismo ejemplo, con una salida gráfica en forma de reloj analógico aparece en el ejemplo de NDS *RealTimeClock* (dentro de *time/RealTimeClock*). El ejemplo es fácilmente transportable a una aplicación de escritorio. por el uso de las funciones de C estándar que se emplean.

## 5 Gestión de la entrada de usuario

Por el amplio uso de estas cuestiones le dedicamos un apartado a este conjunto de operaciones que son muy particulares del hardware de este dispositivo y que permiten acceder a un buen número de sensores de la consola.

La información a este respecto se guarda en un rango de memoria que es compartido entre las aplicaciones para obtener la información del interfaz de entrada de la consola a través del servicio HID. Este se implementa en el fichero *hid.h* y hace referencia a los contenidos de la región de memoria<sup>7</sup>, de solo lectura, que se denomina HID y que es compartida entre todas las aplicaciones.

Ahora nos ocuparemos de la lectura del estado del interfaz de interacción con el usuario (la botonera y la pantalla táctil). Además, dentro de este conjunto de datos de entrada también está el control de volumen (accesible con la función *HIDUSER\_GetSoundVolume*) y otros sensores de entrada de datos, en concreto a los que se obtienen del acelerómetro (*hidAccelRead*) o del giroscopio (*hidGyroRead*).

El ejemplo para 3DS *read-controls* ( en *input/read-controls*) al ejecutarlo nos dará la información de qué teclas han sido pulsadas, están siendo mantenidas pulsadas o se han soltado, como se puede ver en la Figura 9.

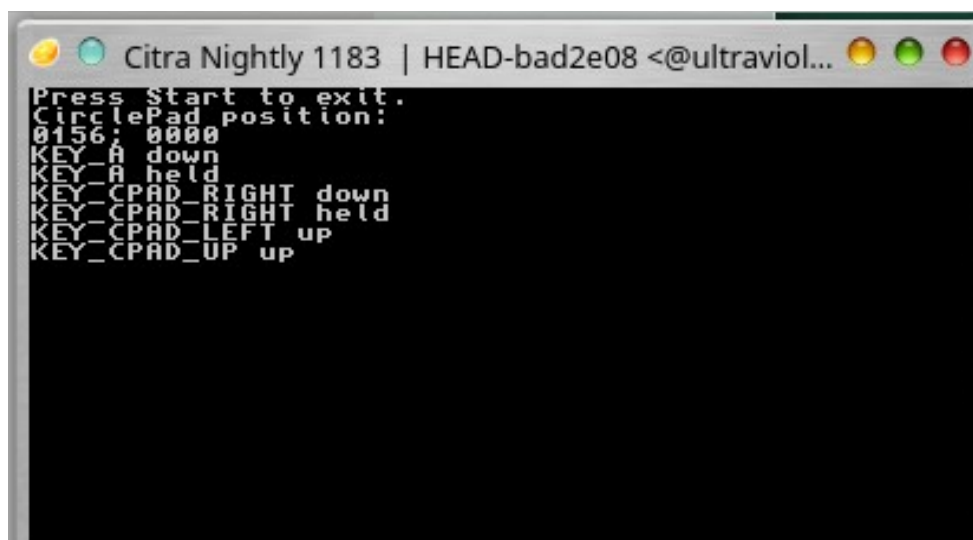
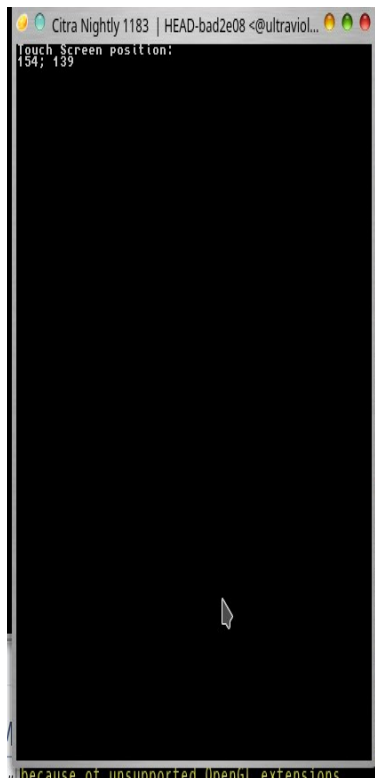


Figura 9: Recorte de la salida en la pantalla superior del ejemplo *read-controls* de la 3DS.

**Ejercicio 7.** Compruebe en el ejemplo de 3DS *read-controls* cómo se comprueban los conjuntos de teclas pulsadas, mantenidas o soltadas en cada iteración del bucle, así como también la función para obtener las coordenadas del *CirclePad*.

<sup>7</sup> Se puede ver el contenido de esta memoria con detalle en la URL [https://www.3dbrew.org/wiki/HID\\_Shared\\_Memory](https://www.3dbrew.org/wiki/HID_Shared_Memory).



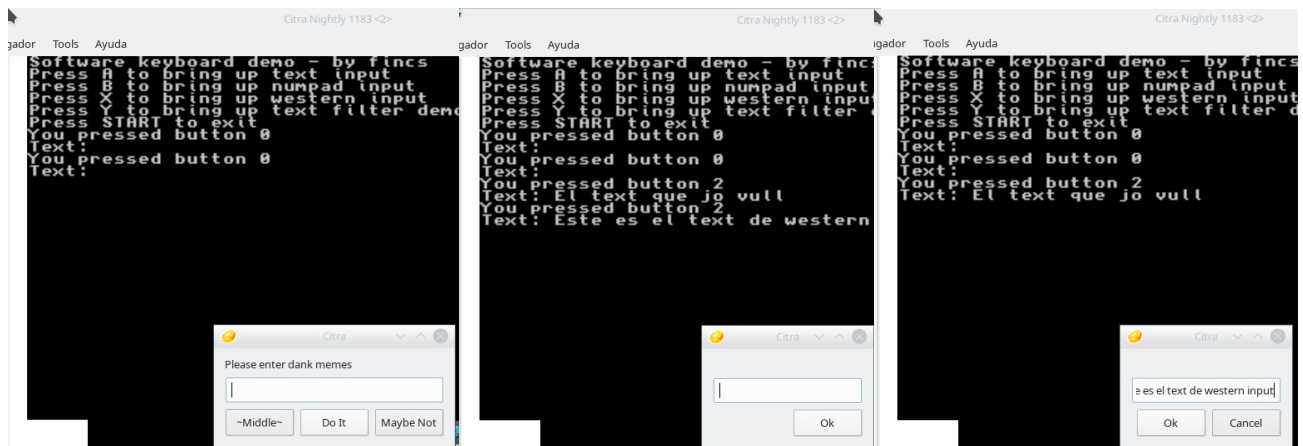
a)



b)

Figura 10: Resultado de ejecutar el ejemplo de 3DS touch-screen: (a) captura completa y (b) detalle del resultado en la pantalla superior.

El ejemplo para 3DS *touch-screen* (en *input/touch-screen*) al ejecutarlo nos proporcionará la información de en qué coordenadas de la pantalla inferior (la táctil) está tocando el lápiz o donde está el cursor del sistema que gestiona el ratón en el emulador, como se puede observar en la Figura 10.



a)

b)

c)

Figura 11: Varios ejemplos de entrada de texto en el ejemplo de 3DS software-keyboard: al pulsar la tecla correspondiente a A (a), a B (B), a X (c) y a D (d).

Un uso interesante de la pantalla táctil es para la gestión de un teclado virtual. Este se puede obtener como en el ejemplo de *software-keyboard* (en *input/software-keyboard*). En el emulador Citra se

verán como cajas de diálogo estándares, véanse las capturas de la Figura 11.

## 6 Sistemas de ficheros: rom vs fat

Para terminar esta práctica queda una cuestión por abordar, cómo se puede realizar la entrada y salida en el sistema de ficheros. Hay dos opciones en el desarrollo de aplicaciones para esta consola: un sistema de solo lectura (denominado *romfs*) y un sistema tanto de lectura como de escritura que está sustentado sobre la tarjeta de memoria<sup>8</sup> del dispositivo (por lo que se denomina *sdmc*) y que utiliza el formato de sistema de archivos FAT.

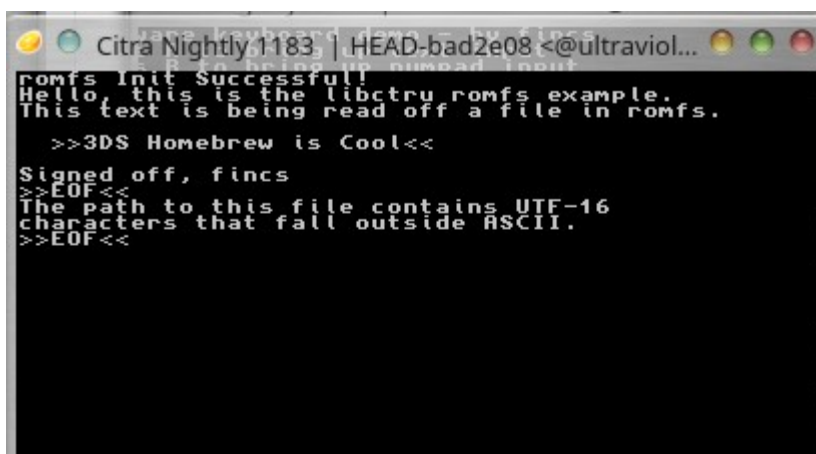


Figura 12: Salida en pantalla del ejemplo de 3DS romfs.

El acceso al sistema de archivos de solo lectura se puede ver en el ejemplo de 3DS *romfs* (dentro de *romfs*, véase Figura 12) y el de acceso a sistemas FAT en *sdmc* (dentro de *sdmc*). En ambos casos, cabe señalar que la secuencia de llamadas al sistema para llevarlos a cabo son utilizables por una aplicación de escritorio.

**Ejercicio 8.** Compruebe el acceso a los ficheros en los dos sistemas de archivos comentados. En el caso del ROMfs, vaya al fichero *file.txt* (dentro del proyecto *romfs/romfs/folder/file.txt*), cambie su contenido y observe como al reconstruir el 3DS y ejecutarlo aparece el texto que haya introducido.

Anote cómo se direcciona el fichero *file.tx* dentro del código y que se pueden usar caracteres Unicode (UTF-16) en la ruta del fichero, como por ejemplo “romfs/romfs/フォルダ/ファイル.txt”.

En el caso del sistema de ficheros sobre la tarjeta de memoria, esta está emulada en<sup>9</sup> Citra en el directorio *\$HOME/.local/share/citra-emu/sdmc/*. Si copia el fichero *test.bin* con, p.ej., la orden

```
$ cp -p sdmc/test.bin $HOME/.local/share/citra-emu/sdmc/
```

quedará como en la Figura 13. Entonces podrá ejecutar esta aplicación y obtener una salida como muestra la Figura 14, cada una se ha obtenido en un instante diferente de la ejecución del ejemplo para 3DS *sdmc*.

**Ejercicio 9.** Compruebe qué ejecuta el ejemplo para 3DS que para que aparezca una imagen estática en la parte inferior, mientras que en la superior se observa un cambio su contenido, como se

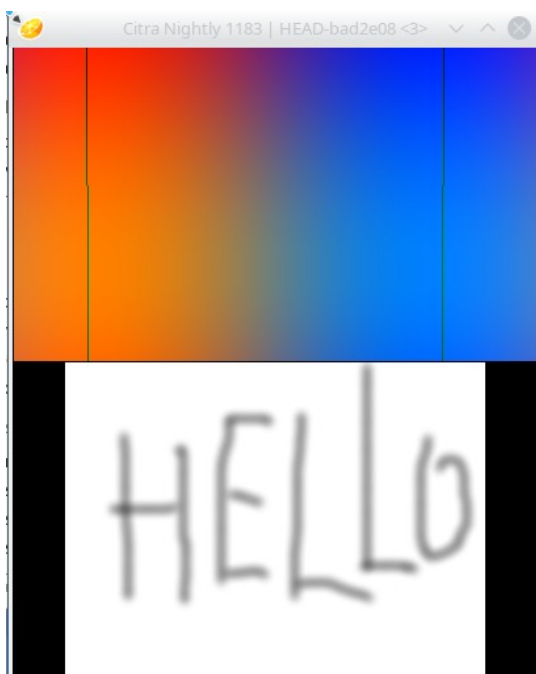
<sup>8</sup> Esta tarjeta de memoria SD se organiza en base a una serie de directorios que se pueden encontrar descritos en *SD Filesystem* <[https://www.3dbrew.org/wiki/SD\\_Filesystem](https://www.3dbrew.org/wiki/SD_Filesystem)>.

<sup>9</sup> El contenido del directorio de usuario, que acoge a la emulación de la tarjeta de memoria SD está explicado en la URL <<https://github.com/citra-emu/citra/wiki/User-Directory>>.

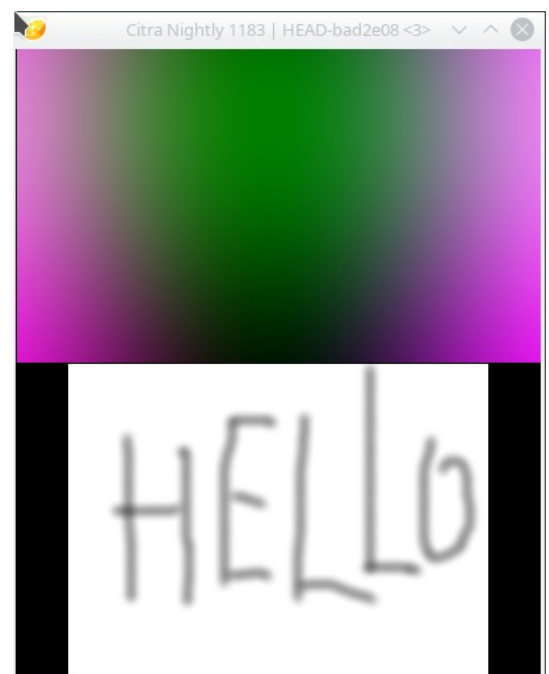
muestra en la Figura 14

|                                       |                                  |
|---------------------------------------|----------------------------------|
| /home/magusti/.local/share/citra-emu/ |                                  |
| Nombre                                |                                  |
|                                       | cheats                           |
| ▼                                     | log                              |
|                                       | citra_log.txt                    |
|                                       | sysdata                          |
| ▼                                     | sdmc                             |
| ▼                                     | Nintendo 3DS                     |
|                                       | test.bin                         |
|                                       | gameover.ogg                     |
|                                       | bonus.ogg                        |
| ▼                                     | 00000000000000000000000000000000 |
| ▼                                     | 00000000000000000000000000000000 |
| ▼                                     | title                            |
| ▼                                     | 00040000                         |
| ▼                                     | 0ff3cc00                         |
| ▼                                     | content                          |
|                                       | 107c5cff.app                     |
|                                       | 00000000.tmd                     |
|                                       | extdata                          |
| ▼                                     | nand                             |
| ▼                                     | data                             |
| ▼                                     | 00000000000000000000000000000000 |

Figura 13: Parte del sistema de ficheros emulado de la consola 3D en el emulador Citra.



a)



b)

Figura 14: Ejemplo de uso de FAT en 3DS: sdmc.

## 7 Trabajo autónomo

Ahora que hemos visto unos cuantos ejemplos es el momento de hacer uno.

Se propone hacer una aplicación para 3DS que cuente el número de clics que el usuario hace en la pantalla táctil durante 3 segundos (aproximadamente) y vaya mostrando el contador en la pantalla superior. Al pulsar la tecla 'A' el proceso se debe reiniciar.

## 8 Bibliografía

- [1] Repositorio de *libctr* - *CTR User Library*. Disponible en la URL <<https://github.com/devkitPro/libctr>>.
- [2] Documentación en línea de *libctr*. Disponible en la URL <<https://libctr.devkitpro.org/>>.
- [3] Colaboradores de Wikipedia. *Código escape ANSI* [en línea]. Wikipedia, La enciclopedia libre, 2013 [fecha de consulta: 19 de febrero del 2014]. Disponible en <[http://es.wikipedia.org/w/index.php?title=C%C3%B3digo\\_escape\\_ANSI&oldid=70047232](http://es.wikipedia.org/w/index.php?title=C%C3%B3digo_escape_ANSI&oldid=70047232)>.
- [3] Repositorio de *citro3d*. Disponible en la URL <<https://github.com/devkitPro/citro3d>>
- [4] Documentación en línea de *citro2d*. Disponible en la URL <<https://citro2d.devkitpro.org/>>.
- [5] Repositorio de *citro2d*. Disponible en la URL <<https://citro2d.devkitpro.org/>>.

## ***Anexo I Documentación en local de libctrú, citro3d y citro2d***

Comentar que se puede obtener la documentación en local de *libctrú* en un PDF a partir del fuente de estas bibliotecas y la aplicación *doxygen* instalada. Para ello, en 2021, hemos ejecutado:

```
$ wget https://github.com/devkitPro/libctrú/archive/master.zip \
-O libctrú-master.zip
$ unzip libctrú-master.zip
$ cd libctrú-master/libctrú
```

Allí hay que editar el fichero *Doxyfile* y cambiar la línea que dice

```
GENERATE_LATEX          = NO
```

por

```
GENERATE_LATEX          = YES
```

y seguimos ejecutando

```
$ doxygen
$ cd docs/latex
$ make
```

Y ya lo tenemos en *libctrú-master/libctrú/docs/latex/refman.pdf*. Podría guardar el fichero junto a los ejemplos o la instalación en `${DEVKITPRO}` o donde quiera. Eso sí, sugiero cambiarle el nombre por *libctrú\_ManualDeReferencia.pdf* (o algo parecido). Acceda a *libctrú - CTR User Library* [1] y compruebe que tiene la misma estructura que lo que acaba de instalar.

Puede hacer las mismas operaciones para la documentación de *citro2d* [4] y *citro3d*<sup>10</sup>, a partir del código fuente de las mismas y repitiendo el proceso anterior, con la salvedad de que habrá de descargarlas de sus repositorios:

- Para el caso de *citro3d* [3] con  
\$ wget https://github.com/devkitPro/citro3d/archive/master.zip -O citro3d-master.zip
- Para el caso de *citro2d* [5] con  
\$ wget https://github.com/devkitPro/citro2d/archive/master.zip -O citro2d-master.zip

Y repitiendo el proceso análogo al realizado para la documentación de la *libctrú*.

---

<sup>10</sup> De hecho, a fecha de la redacción de esta práctica, no está publicada la documentación para *citro3d*. Vease <<https://devkitpro.org/viewtopic.php?t=8857>>: “citro3d documentation on the other hand doesn't exist; like you pointed out only examples are available”.