

Arquitectura y Entornos de desarrollo para Videoconsolas

Práctica 1

Instalación del entorno de desarrollo *devkitPro* para la arquitectura de la videoconsola NDS y N3DS

La práctica se realizará en entorno GNU/Linux en el laboratorio. Vamos a abordar la instalación y comprobación de las herramientas necesarias para realizar desarrollos en la NDS y N3DS.

Recuerde que se le pedirá que guarde resultados de sus acciones a lo largo de las prácticas para confeccionar el portfolio, que es una parte de las prácticas. No descuide pues hacer copias de lo que necesite en su espacio del servidor de la asignatura. Para ver los detalles de cómo acceder a las cuentas personales desde cualquier máquina diferente de las del laboratorio, consulte el documento en

PoliformaT Aev: Recursos Prácticas / Prácticas / Acceso a la cuenta de usuario.pdf

Dada la situación continuada de docencia en línea en el curso 2020/2021, se ha portado la imagen de la configuración del laboratorio (Ubuntu 18.04. LTS) a una imagen para ser recreada como máquina virtual sobre VirtualBox. Los detalles de cómo instalarla en tu propio equipo y cómo instalar el emulador de la 3DS para las librerías disponibles en esta distribución se encuentran en el documento:

PoliformaT Aev: Recursos Prácticas / Prácticas / presentacioMaquinaVirtual_2k20_2k21.pdf

1 Instalación de *devkitPro*

devkitPro constituye la pieza central del desarrollo con un SDK no oficial disponible. Es multiplataforma y proporciona las herramientas y las librerías básicas para el desarrollo. También ofrece un conjunto de librerías externas recompiladas (portadas) a algunas de las plataformas de desarrollo para facilitar el uso de formatos de ficheros de medios, el uso de operaciones de alto nivel de manejo de gráficos y sonido, acceso a ficheros XML o comprimidos, etc.

Para su uso en el laboratorio ya se encuentra instalado, pero si ha de ser utilizado en una máquina propia puede proceder a instalarlo tomando como ejemplo la lista de instrucciones seguidas para la instalación del laboratorio que se encuentran en el Anexo A. Proceso de instalación de *devkitPro* utilizado en el laboratorio de prácticas de AEV.

Para comprobar rápidamente que puede proceder con el resto de la práctica, abra un terminal y ejecute la orden

```
$ set | grep devkitpro
```

Debería obtener un resultado del estilo de

```
DEVKITARM=/opt/devkitpro/devkitARM
```

```
DEVKITPPC=/opt/devkitpro/devkitPPC
```

```
DEVKITPRO=/opt/devkitpro
```

En caso afirmativo, podrá comprobar con la orden

```
$ ls -l $DEVKITPRO
```

que el directorio que indica la variable de sistema *DEVKITPRO* contiene, al estilo de lo que muestra

el listado 1, tanto el instalador (*pacman*), como las versiones de las herramientas de desarrollo (*devkit**), librerías (*lib**) y, sobre todo, los ejemplos (*examples*) de partida. Veamos como reconstruirlos en los siguientes apartados.

```
$ ls -l $DEVKITPRO
total 132K
drwxr-xr-x    8 magusti magusti 4,0K sep  5 13:16 pacman
drwxr-xr-x    3 root      root   4,0K sep  5 13:25 tools
-rw-r--r--    1 root      root   270 ene  1 22:22 wii.cmake
-rw-r--r--    1 root      root   537 ene  1 22:22 switchvars.sh
-rw-r--r--    1 root      root   531 ene  1 22:22 switch.cmake
-rw-r--r--    1 root      root   334 ene  1 22:22 ppcvars.sh
-rwxr-xr-x    1 root      root   199 ene  1 22:22 portlibs_prefix.sh
-rw-r--r--    1 root      root   405 ene  1 22:22 ndsvars.sh
-rwxr-xr-x    1 root      root  1,5K ene  1 22:22 meson-toolchain.sh
-rwxr-xr-x    1 root      root   502 ene  1 22:22 meson-cross.sh
-rw-r--r--    1 root      root   280 ene  1 22:22 gamecube.cmake
-rw-r--r--    1 root      root   306 ene  1 22:22 devkitppc.sh
-rw-r--r--    1 root      root   421 ene  1 22:22 devkitppc.cmake
-rw-r--r--    1 root      root   295 ene  1 22:22 devkitarm.sh
-rw-r--r--    1 root      root   424 ene  1 22:22 devkitarm.cmake
-rw-r--r--    1 root      root   270 ene  1 22:22 devkita64.sh
-rw-r--r--    1 root      root   433 ene  1 22:22 devkita64.cmake
-rw-r--r--    1 root      root   439 ene  1 22:22 3dsvars.sh
-rw-r--r--    1 root      root   254 ene  1 22:22 3ds.cmake
drwxr-xr-x    4 root      root   4,0K ene 10 11:57 libctr
drwxr-xr-x    2 root      root   4,0K ene 10 11:57 cmake
drwxr-xr-x   11 root      root   4,0K ene 10 11:57 devkitPPC
drwxr-xr-x    4 root      root   4,0K ene 10 11:57 libogc
drwxr-xr-x    9 root      root   4,0K ene 10 11:57 examples
drwxr-xr-x    4 root      root   4,0K ene 10 11:57 libmirko
drwxr-xr-x    4 root      root   4,0K ene 10 11:57 libnx
drwxr-xr-x    4 root      root   4,0K ene 10 11:57 libtonc
drwxr-xr-x    4 root      root   4,0K ene 10 11:57 libgba
drwxr-xr-x    4 root      root   4,0K ene 10 11:57 libnds
drwxr-xr-x    6 root      root   4,0K ene 10 11:57 portlibs
drwxr-xr-x    8 root      root   4,0K ene 10 11:58 devkitA64
drwxr-xr-x    8 root      root   4,0K ene 10 11:58 devkitARM
drwxr-xr-x   16 root      root   4,0K ene 10 11:58 licenses
```

Listado 1: Conjunto de ficheros utilizados durante la instalación de devkitARM_r46, con libnds 1.6.1 y libctr 1.2.0 .

2 Ejemplos de *devkitPro* para la plataforma NDS

En la última versión del instalador automático se ha incorporado la instalación de los ejemplos de la biblioteca de funciones *libnds* ([2] y [3]). Cópielos a un directorio de su espacio de usuario para poder reconstruirlos y modificarlos, p. ej., utilizando la orden

```
$ cp -rp ${DEVKITPRO}/examples/nds desarrollo_NDS
```

Para poder crear las versiones ejecutables de todos ellos utilizaremos los ficheros *Makefile* que vienen creados al efecto y podremos recompilarlos todos con

```
$ cd desarrollo_NDS
$ make
```

Para ejecutarlos, de entre los diferentes emuladores existentes¹, haremos uso de un emulador. Actualmente *DeSmuME* y *no\$gba* están disponibles en el laboratorio. Usaremos el primero por facilidad de instalación ya que está en el repositorio de la distribución de Linux que se utiliza. Se puede llevar a cabo desde la línea de órdenes con:

```
$ sudo apt-get install desmume
```

2.1 Emuladores NDS

DesMuME está presente en la instalación del laboratorio. Se puede ejecutar una aplicación para la NDS con una orden, como p. ej.;

```
$ desmume hello_world/hello_world.nds
```

También se podría un segundo emulador, aunque solo se cita aquí por curiosidad histórica. El *no\$gba*, que está sólo en versión de ejecutable para plataforma MS/Windows. Para instalar el emulador *no\$gba*. Existen dos versiones disponibles, cualquiera de ellas nos sirve, aunque se sugiere escoger la primera:

1. La del autor (v2.9b - 30 Sep 2018), desde la página "no\$gba Gameboy Advance / Nintendo DS / DSi Emulator Homepage" en la dirección <http://problemkaputt.de/gba.htm>
2. La versión *no\$gba Freeware Version* (v2.6a) que se puede descargar de algunos repositorios de software en Internet.

A partir de cualquiera de estas dos versiones que haya descargado, descomprima el fichero y, en Linux, ejecute el instalador con *Wine*:

```
$ unzip no\$gba-w.zip
$ wine NO\$GBA.EXE
```

Con *DeSmuMe* es posible depurar una aplicación al tiempo que la vemos en ejecución. En el

¹ Nintendo 3DS emulators. http://emulation-general.wikia.com/wiki/Nintendo_3DS_emulators.

Anexo B. Depuración de programas en NDS hay más información a este respecto aunque, de momento, no entraremos a utilizarla.

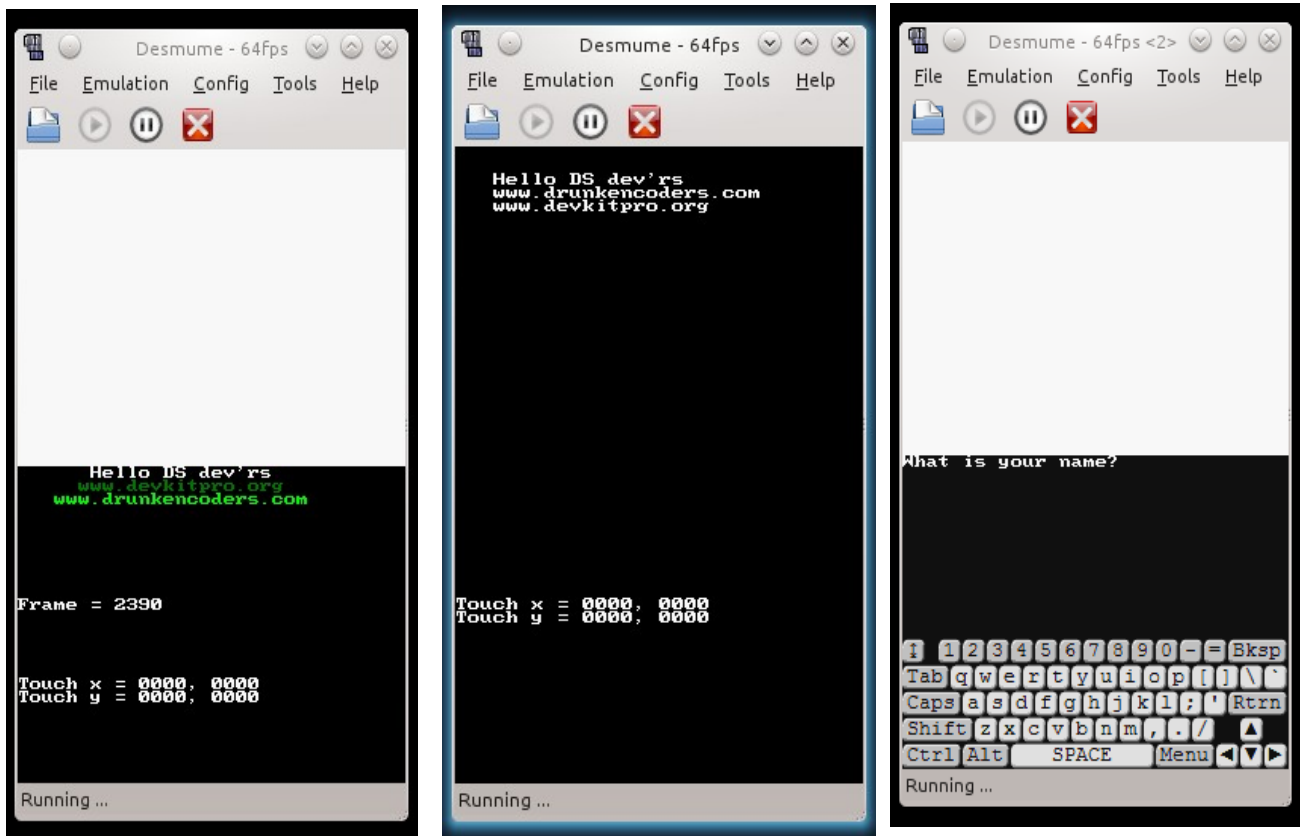


Figura 1: Capturas de los ejemplos `hello_world.nds`, `print_both_screens.nds` y `keyboard_stdin.dsv`.

Llegados a este punto, pruebe a ejecutar alguno de los resultados obtenidos, la Figura 1 muestra el resultado de ejecutar²

```
$ desarrollo_NDS/hello_world/hello_world.nds
$ desarrollo_NDS/Graphics/Printing/print_both_screens/print_both_screens.nds
$ desarrollo_NDS/input/keyboard/keyboard_stdin/keyboard_stdin.nds
```

3 Ejemplos de *devkitPro* para la plataforma 3DS

En la última versión del instalador automático se ha incorporado la instalación de los ejemplos de la biblioteca de funciones *libctru* ([8] y [9]). Esta también se apoya en *devkitARM* y sus autores la describen con un nivel intermedio entre el soporte de ejecución de la N3DS y un posible *port* de SDL. Cópie los ejemplos a un directorio de su espacio de usuario para poder reconstruirlos y modificarlos, p. ej., utilizando la orden

```
$ cp -rp ${DEVKITPRO}/examples/3ds desarrollo_3DS
```

Para poder crear las versiones ejecutables de todos ellos utilizaremos los ficheros *Makefile* que

²Pruebe también con `fire_and_sprites.nds`.

vienen creados al efecto y podremos recompilarlos todos con

```
$ cd desarrollo_3DS
```

```
$ make
```

En alguna instalación aparece un problema en uno de los ejemplos que interrumpe la generación del resto. En ese caso y solo en ese caso, hay que modificar el fichero *Makefile*, donde dice

```
@for i in $(MAKEFILES); do $(MAKE) -C `dirname $$i` || exit 1; done;
```

podemos dejarlo como

```
@for i in $(MAKEFILES); do $(MAKE) -C `dirname $$i`; done;
```

y ya podemos volver a ejecutar el *make*. Para ejecutarlos haremos uso de un emulador, **en el laboratorio es necesario instalarlo**. Veamos qué hay que hacer para ello

3.2 Emuladores N3DS

De entre los diferentes emuladores existentes, véase la Figura. 2, para la 3DS escogemos Citra [6], véase Figura 3, por sus características multiplataforma (Linux, Windows y OS X) y por su nivel de características emuladas de la plataforma original.



Name	Operating System(s)	Latest Version	Active	Recommended
3dmoo	Windows	Git	✓	X
Citra	Windows / OS X / Linux	Git	✓	X
TronDS	Windows	v1.0.0.5	✓	X

The Nintendo 3DS currently has three emulators in early development which are Citra, 3dmoo, and TronDS. Citra is capable of running homebrew games. Most of hardware has yet to be documented. Once the console has been hacked (to some degree, at least), and the hardware is well known and its functions are documented, emulation can begin. Plus the fact that the 3DS supposedly has a high amount of security built-in, this may take a while. Be prepared to wait.

Figura 2: Comparativa de emuladores disponibles para 3DS.

Sus responsables no dudan en remarcar *el perfil experimental de Citra*, solo lleva desde 2014. Necesita ejecutarse sobre un sistema operativo de 64 bits y en el que haya una versión de *OpenGL* 3.3 o superior con el que los gráficos son renderizados en software. Y vuelven a destacar que ya soporta aplicaciones que hacen uso del audio³, las cámaras y algunas funciones de acceso a la red.

3 MerryMage (May 19th, 2016). *HLE Audio Comes to Citra*.



Figura 3: Logotipo del emulador de 3DS Citra y informes de desarrollo [6] .

Nightly Build Last release was 2 days ago





















Build Date	Commit Information	Download
2 days ago	Nightly Build - 6b2e7b7	   
2 days ago	Nightly Build - e2adb51	   
2 days ago	Nightly Build - a1d6396	   
3 days ago	Nightly Build - 2461f67	   
3 days ago	Nightly Build - 2539215	   

Figura 4: Ejemplo de la página web de descarga de Citra .

Desde la pagina web del proyecto [6] tenemos acceso a las versiones de instalación en el apartado “Download”. Desde el enlace [Manual download](#) se pueden descargar diferentes binarios o el código fuente y recompilarlo. Escogeremos la más reciente de las versiones precompiladas del apartado *Nightly Build*, véase *Figura 4*, y que se descarga haciendo pinchando sobre el icono de *Tux*.

Tras descargarlo se deberá descomprimir y lo “instalaremos” en nuestra cuenta de usuario con:

```
$ mkdir -p ~/bin/
$ tar xvf citra-linux-20210204-6b2e7b7.tar.xz -C ~/bin/
$ echo '~/bin/citra-linux-20210204-6b2e7b7/citra-qt $*' >> ~/bin/citra-qt.sh
$ chmod +x ~/bin/citra-qt.sh
$ echo '~/bin/citra-linux-20210204-6b2e7b7/citra $*' >> ~/bin/citra.sh
$ chmod +x ~/bin/citra-qt.sh ~/bin/citra.sh
$ export PATH=$PATH:~/bin
```

Tenga en cuenta que la última orden deberá copiarla en el fichero `~/.bashrc` para que cada vez que inicie la sesión se actualice la variable `PATH`. Y a partir de ahora ya deberíamos poder ejecutar cualquiera de las dos versiones que obtenemos de *citra*, con el resultado que aparece en la fig. 5 obtenido a partir de la ejecución de la orden:

```
$ citra-qt.sh
```



Figura 5: Inicio del emulador Citra .

Ahora ya podemos probar los ejemplos en formato 3DSX de *desarrollo_3DS* cargándolos con el men *File | Load File ...*. También se puede pasar la ruta al fichero como argumento de los ficheros shell-script (*.sh) que hemos creado.

Compruébelo con ejemplos que se muestran en las figuras 6, 7, 8 y 9: *read-controls*, *romfs*, *fragment_light*, o *textured_cube*. Pruebe también con el ejemplo de *gpusprites*.



Figura 6: Ejemplo read-controls.3dsx.

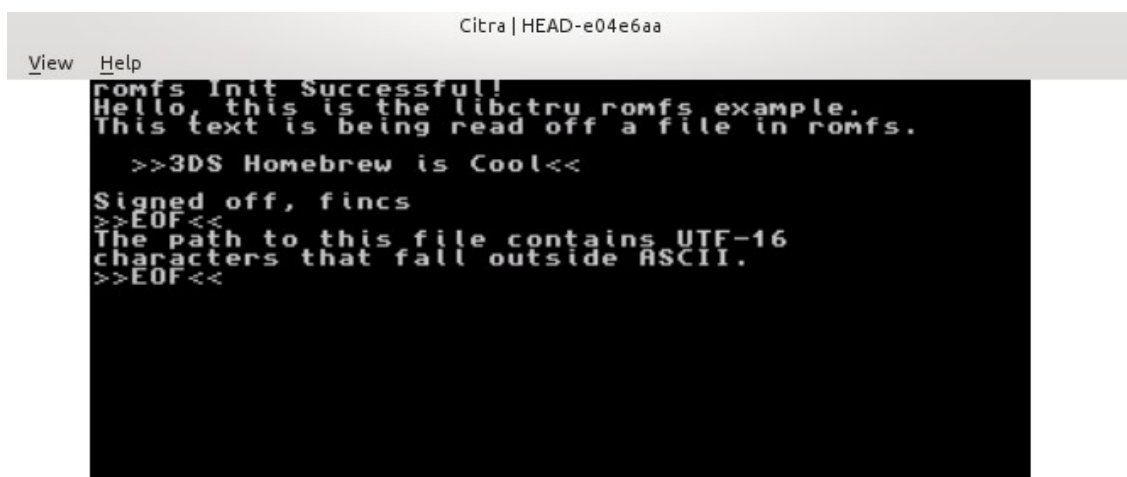


Figura 7: Ejemplo romfs.3dsx.

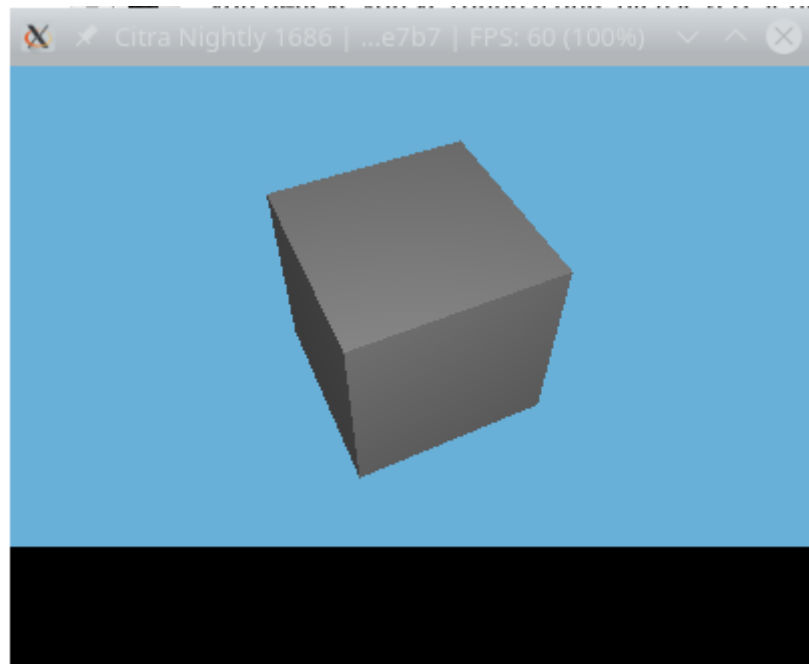


Figura 8: Ejemplo fragment_light.3dsx.

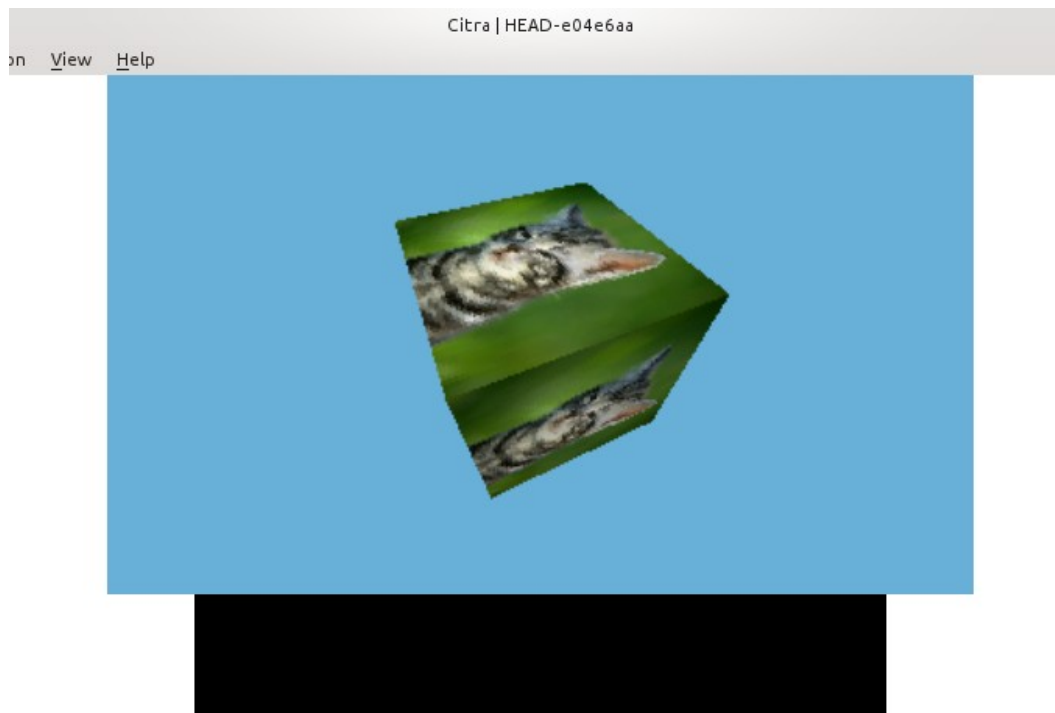


Figura 9: Ejemplo textured_cube.3dsx.

4 Trabajo autónomo

En función de la disponibilidad de videoconsolas reales, en el laboratorio se ofrecerá el acceso a las mismas, realice una copia de los resultados y pruebe a ejecutarlos en ellas. En general el desarrollo se realizará con emuladores, lo que permite un desarrollo rápido y la toma fácil de capturas de pantalla de la ejecución del mismo.

Actividad 1. Como primera parte de los resultados a obtener de esta práctica y **en un equipo diferente de los del laboratorio**, se procederá a instalar las herramientas de desarrollo de *devkitPro* y realizar alguna captura de pantalla que muestre, junto a la imagen de quien lo realiza, los siguientes momentos de la misma:

1. La ventana donde se está realizando la instalación con *dkp-pacman*.
2. La ventana donde se están compilando los ejemplos para la plataforma NDS.
3. La ventana donde se están compilando los ejemplos para la plataforma 3DS.
4. *DeSmuME* ejecutando los ejemplos de la plataforma NDS *simple* y *animate_simple*.
5. *Citra* ejecutando los ejemplos de la plataforma 3DS *gpusprites* y *colored-text*.

Actividad 2. Como segunda parte de los resultados a obtener de esta práctica se procederá a llevar a cabo el desarrollo de un proyecto complejo que se llama “How to Make a Bouncing Ball Game”⁴. El trabajo de Jose David [5] ofrece información complementaria para ampliar varios aspectos de la funcionalidad de este ejemplo y queda como material de ampliación de trabajo con la NDS. Dados los cambios del SDK de devKitPro se ha hecho necesario revisar toda la implementación y se puede acudir al contenido del *github* en [10]

Para documentar esta actividad, se deberán realizar capturas de pantalla que muestren hasta dónde se ha podido completar de los siguientes tres pasos:

1. Descargar, compilar y ejecutar el resultado del proyecto con el emulador DeSmuME.
2. Ampliar el ejemplo con la creación de otros 50 *sprites* que sean copia del único que existe ahora (la pelota), incorporando al ejemplo el ejemplo del tutorial original (guardado en PDF junto con el enunciado de la práctica). El movimiento de estos cincuenta elementos debe ser aleatorio.
3. Detectar las colisiones de la pelota original con las 50 réplicas y, en caso de que se de, hacer desaparecer la réplica “golpeada”.
4. Poner mensajes de texto de inicio (carga de recursos), avance del juego (número de pelotas que se han hecho desaparecer) y cierre (nombre de el/los autor/es).

⁴ El ejemplo original [4] es del autor de la parte de sonido (*Maxmod*) del SDK no oficial y cuyo tutorial original está guardado en un PDF, junto con los elementos gráficos en el apartado de esta práctica en el *PoliformaT* de AEV. Este tutorial está incompleto, por eso no es la referencia principal para llevarlo a cabo.

5 Bibliografía y referencias

- [1] *devkitPro*. URL: <<http://sourceforge.net/projects/devkitpro/>>
- [2] Libnds. URL: <<http://sourceforge.net/projects/devkitpro/files/libnds/>>
- [3] Libnds Documentation. URL: <<http://libnds.devkitpro.org/>>
- [4] Mukunda Johnson. “How to Make a Bouncing Ball Game” <<http://ekid.nintendev.com/bouncy/>>. Disponible en el *PoliformaT* de esta asignatura *Recursos > Prácticas practica1_instalacion > How_to_Make_a_Bouncing_Ball_Game.pdf*
- [5] J. D. Jaén. (2015). Tutorial práctico para desarrollo de videojuegos sobre plataforma Nintendo NDS. Disponible en el *PoliformaT* de esta asignatura *Recursos > Materiales para el trabajo de asignatura > jjJaen_TutorialPracticoParaDesarrolloDeVideojuegosSobrePlataformaNintendoNDS.pdf*
- [6] Citra. URL <<https://citra-emu.org/>>.
- [7] M. J. Santofimia y F. Moya. (2010). Laboratorio de Estructura de Computadores empleando videoconsolas Nintendo DS. ISBN 978-84-9981-039-3. Disponible en <<https://www.bubok.es/libros/190123/Laboratorio-de-Estructura-de-Computadores-empleando-videoconsolas-Nintendo-DS>>.
- [8] libctru – CTR User Library. URL <<https://github.com/smealum/ctrulib>>.
- [9] Documentación de libctru. URL <<https://libctru.devkitpro.org/>> y <<https://devkitpro.github.io/libctru/>>.
- [10] M. Agustí. (2020). NDS-homebrew-development. URL <<https://github.com/magusti/NDS-homebrew-development>>.

6 Anexo A. Proceso de instalación de *devkitPro* utilizado en el laboratorio de prácticas de AEV

En Febrero de 2020, aquí se propone cómo llevar a cabo la instalación para la plataforma GNU/Linux *Ubuntu 18.04 LTS* y *Ubuntu 20.04.02 LTS*. Para otras plataformas se puede encontrar la información al respecto en <https://devkitpro.org/wiki/Getting_Started>, así como para *docker* <<https://hub.docker.com/u/devkitpro>>.

Nota: acompaño las órdenes con el prompt (“\$”) para indicar que hay que realizarlas en un terminal y con "sudo" para lo que requiere permisos de administrador (como "super") y lo que no lo lleva, es que lo debe poder hacer el usuario, como tal.

Para la instalación en Linux ahora hay que utilizar el instalador de paquetes *Arch Linux pacman* <<https://wiki.archlinux.org/index.php/pacman>> (no confundir con el juego). Para ello: hay que instalar devkitPro pacman 1.0.1:

```
$          wget          https://github.com/devkitPro/pacman/releases/download/v1.0.2/devkitpro-pacman.amd64.deb
$ sudo dpkg -i /tmp/devkitpro-pacman.amd64.deb
Y aparecerá la versión del instalador pacman para devkitpro en /opt
$ ls -l /opt/devkitpro/
```

Ahora ya es posible instalar la última versión de *devkitPro*, para lo que hay que ejecutar:

```
$ sudo dkp-pacman -U https://downloads.devkitpro.org/devkitpro-keyring.pkg.tar.xz
$ sudo dkp-pacman -Sy
$ sudo dkp-pacman -Sya
$ sudo dkp-pacman -Sl | while read linea; do cut -d" " -f2; \
done | tee totesLesLlibreries.txt
$ sudo dkp-pacman -S `cat totesLesLlibreries.txt`
```

Y podemos aprovechar para instalar DeSmuME:

```
$ sudo apt-get install desmume
```

Para comprobar el funcionamiento, deberían estar declaradas las variables durante la instalación, pero habría que actualizar las variables de entorno ejecutando la orden

```
$ source /etc/profile.d/devkit-env.sh
```

que deberá ser incorporada en el perfil de cada usuario automáticamente o manualmente en el *~/.bashrc* de cada usuario. Este *script* se encarga de declarar las variables respecto a donde se ha instalado *devkitPro* y, dado, que en la versión actual falta la declaración de DEVKITA64 se sugiere incluir estas líneas en el *~/.bashrc* de cada usuario:

```
export DEVKITPRO=/opt/devkitpro
export DEVKITARM=${DEVKITPRO}/devkitARM
export DEVKITPPC=${DEVKITPRO}/devkitPPC
export DEVKITA64=${DEVKITPRO}/devkitA64
export PATH=${DEVKITPRO}/tools/bin:$PATH
```

7 Anexo B. Depuración de programas en NDS

Es posible depurar las aplicaciones de NDS al tiempo que ir viendo como avanza la aplicación en el emulador ya que uno y otro admite opciones para recibir y enviar ordenes relativas a este proceso a través de puertos locales del computador [7]. Para ello hay que:

- Revisar que el fichero *Makefile* utilizado tenga el parámetro **-g** entre la lista de los *CFLAGS*. De no ser así, habría que incluirlo.
- Disponer de una versión del depurador *gdb* instalado. Se sugiere utilizar uno con salida gráfica por que facilita la identificación de las órdenes al depurador con iconos gráficos. Por ejemplo se puede utilizar *kdbg*⁵ que está disponible en los repositorios de las distribuciones habituales de *GNU/Linux*.

Realizadas estas comprobaciones ya tendremos con *kdbg* un interfaz para lanzar la depuración del ejecutable mientras se visualiza el código fuente y, por otro lado, *DeSmuME* para emular la ejecución de la aplicación y así poder ver su avance.

Si tenemos estos dos elementos, sera posible ejecutarlos desde la linea de órdenes o escribir un *script* que lo haga por nosotros. En cualquier caso, hay que conseguir que ambas aplicaciones (*desmume* y *kdbg*) se ejecuten y se puedan comunicar. Para ello escogeremos un número de puerto y les indicaremos a ambas aplicaciones que lo utilicen para comunicarse entre ellas.

Puede ser el 7777 o cualquier otro que esté libre en nuestro equipo. Para el ejemplo de *hello_world* que acompaña a la distribución de *DevkitPro*, se puede hacer con la secuencia de órdenes que se muestra y donde se han suprimido muchos de los mensajes, para solo dejar los que son pertinentes en este momento:

```
$ desmume --arm9gdb=7777 hello_world.nds &
```

```
...
```

```
DeSmuME 0.9.11 svn0 dev+ x64-JIT
```

```
Created GDB stub on port 7777
```

```
STALL
```

```
...
```

```
Processing packet g
```

```
'g' command PC = 00000000
```

```
...
```

```
$ kdbg -r :7777 hello_world.elf
```

Con esta secuencia y abriendo el fichero fuente *hello_word.c.*, la fig. 10, muestra un instante de la depuración del ejemplo *hello_world*. Cabe observar que:

⁵ *Kdbg. A Graphical Debugger Interface..* Disponible en <<http://www.kdbg.org/>>.

1. El *Makefile* contiene el parámetro -g.
2. *DesmuMe* está ejecutándose como cabe esperar de él, esto es muestra la salida de la aplicación al tiempo que permite la interacción del usuario con la emulación de la pantalla táctil, indicando la posición donde se actúa con el cursor a modo de puntero de la consola.
3. Kdbg muestra el código fuente, permitiendo controlar la ejecución de las instrucciones del mismo.
4. También sería posible ver el código ensamblador de ARM que se genera a partir del código C activando el símbolo '+' junto a una instrucción concreta del código fuente original.
5. Es posible insertar puntos de ruptura pinchando con el ratón en el espacio a la izquierda del símbolo '+'. Apareciendo, en ese caso, unos marcadores en forma de pequeñas esferas de color rojo.
6. Mientras se ejecuta el código es posible ver el valor de todas las variables locales en el área de “Locales” y de las que queramos explícitamente en la de “Expresiones vigiladas”.

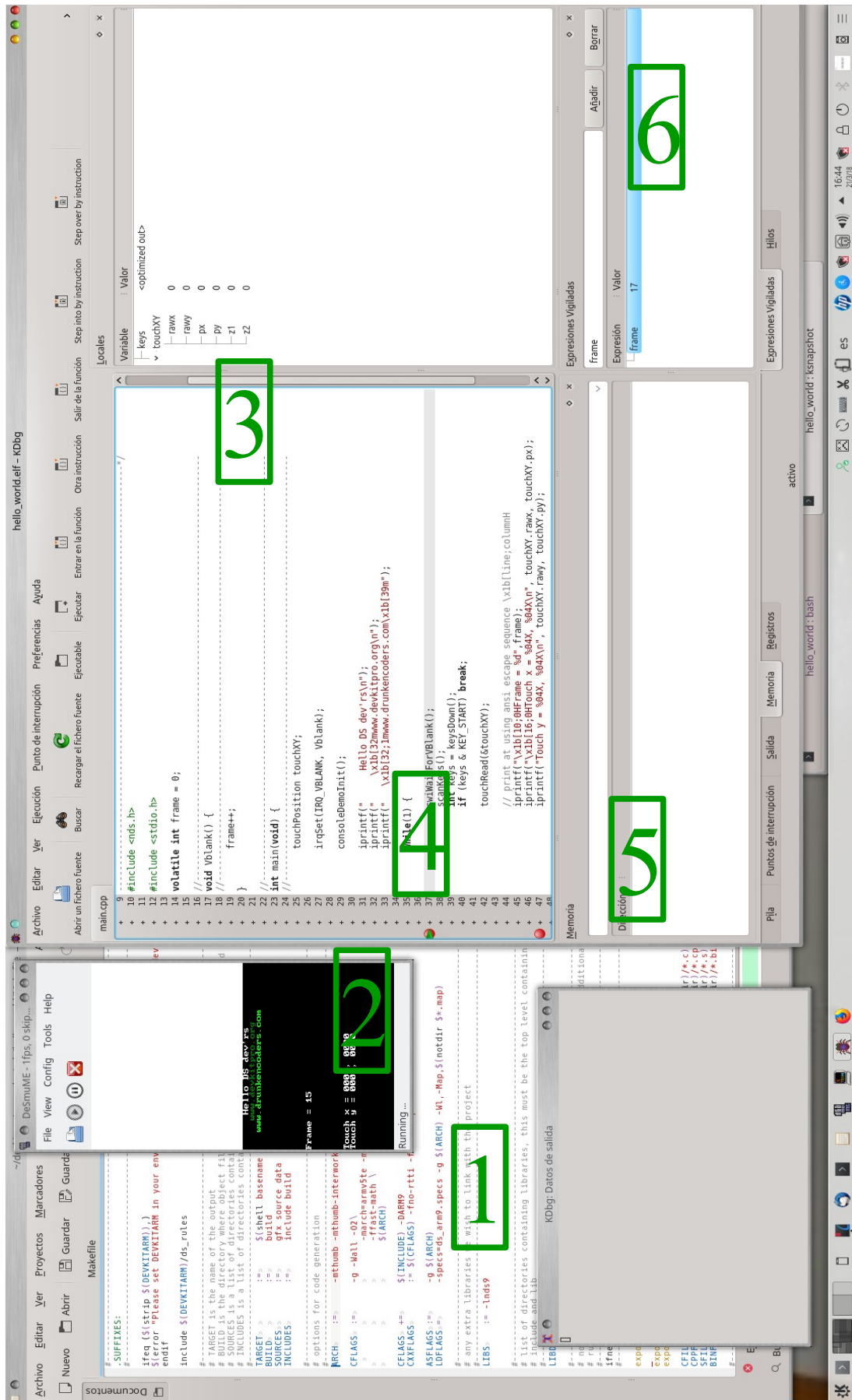


Figura 10: Depurando hello_world.nds.