

Capítulo 1

Nuestra primera aplicación

Este es el típico capítulo de cualquier libro de programación, donde se explica como mostrar una cadena de texto por la pantalla siguiendo todo el proceso de creación, compilación y ejecución de nuestro código.

1.1. ¿Cómo lo programo? ¿Cómo creo una ROM?

Antes de nada, debemos saber como hemos de programar y con qué. Para programar para la Nintendo DS, utilizaremos varias cosas. En primer lugar **DevKitPro** (<http://www.devkitpro.org/>), que es un conjunto de librerías, compiladores y utilidades para desarrollar software para varias plataformas, entre ellas NDS, PSP, GameCube, GP32,... es libre y de descarga gratuita. DevKitPro viene con las librerías **libnds** que son las librerías que adaptan el código C/C++ que realicemos al hardware específico de la NDS. Además utilizaremos **PALib** (<http://www.palib.com/>), un conjunto de librerías de mayor nivel que las libnds, es decir, facilitan su uso ya que en el fondo PALib utiliza las libnds, además de aportar algunas utilidades como PAGF para la conversión de los gráficos. Como entorno de desarrollo pueden utilizar cualquier editor de textos, y luego ejecutar el compilador, aunque hay templates para Programmers Notepad y VHAM, en dichas webs se pueden encontrar instrucciones de como adaptar tu IDE preferido al desarrollo con DevKitPro. No se dan más especificaciones de cómo realizarlo y se deja a cargo del lector, ya que según el sistema operativo y el IDE usado difieren unas de otras.

Resumiendo, programaremos en C o C++, usando PALib y DevKitPro.

1.2. Plantillas al descubierto

Lo mejor a la hora de empezar a programar una aplicación o juego es utilizar una plantilla, para un proyecto de este tipo son varios los archivos y carpetas necesarios, así como mantener una estructura para que el compilar pueda encontrar todos los archivos necesarios. Una plantilla es una forma rápida de crear todo esto, es una estructura que contiene todos los archivos de desarrollo necesarios para el correcto funcionamiento del proyecto, dentro de esa estructura es donde iremos añadiendo nuestros fondos, imágenes, sonidos, archivos de código...

1.2.1. Directorios

- *source*, contiene los archivos *.c* o *.cpp* que son archivos de código, también incluirá una subcarpeta *gfx* con los archivos de gráficos, pero de eso hablaremos más tarde.
- *include*, contiene los archivos de cabecera, cuya extensión es *.h* y que por el momento no trataremos con ellos.
- *data*, ahí alojaremos los sonidos, música,... algo que veremos mucho más adelante.
- *build*, adicionalmente y una vez que compilemos se creará un nuevo directorio en el que se alojarán archivos auxiliares para la compilación, no tendremos que tocar nada de aquí.

1.2.2. Archivos

- *Makefile*, es el archivo encargado de explicarle al compilador cómo ha de hacer su trabajo. Muy importante, pero no será necesario modificarlo.
- *logo.bmp* y *logo.wifi.bmp*, son las imágenes correspondientes a nuestro juego, que se enviarán si jugamos via Wifi o si disponemos de un menú con iconos (como M3).
- *build.bat*, se encarga de compilar nuestra aplicación.
- *clean.bat*, borrará la carpeta *build* y todo su contenido así como las *roms*, sirve para limpiar y empezar "de cero". Con el uso de *build.bat* se crearán archivos que quizás en nuestra versión final no utilicemos o que nos generen errores, limpiándolos conseguiremos resetear el proceso de compilado e incluso reducir el tamaño de nuestra *rom*. En general no es necesario utilizarlo, a menos que el compilador nos dé algún problema sobre referencias de archivos o antes de hacer pública nuestra aplicación, así tendrá el tamaño menor posible y no contendrá nada innecesario.
- *project.vhw* Plantilla para VisualHam.
- *Template.pnproj* Plantilla para Programmers Notepad.
- *source/main.c* Dentro de la carpeta *source*, este archivo es el que contendrá el código de nuestro programa principal.

Para crear un nuevo programa sólo tendrá que situarse en el directorio donde instaló DevKitPro, copiamos la carpeta *PALib Template* y la renombramos con un nuevo nombre, por ejemplo *NuevoPrograma*. Una vez hecho esto accedemos a ella (carpeta *NuevoPrograma*), y hacemos doble click en el archivo *Template.pnproj*, esto abrirá el Programmers Notepad (de ahora en adelante PN), que es el que utilizaremos como entorno de desarrollo a lo largo de este libro. También podemos abrir *project.vhw* para utilizar VHAM como entorno de desarrollo o cualquier otro programa debidamente configurado. Elegimos PN debido a su simplicidad, su poco gasto de recursos y sus altas prestaciones, lo mismo sirve para compilar C, PHP o escribir ficheros de texto plano. Una vez abierto PN, seleccionamos del menú-árbol de la izquierda el fichero *main.c*. Para

compilar nuestro código fuente, en el Programmers Notepad, haremos uso de la combinación de teclas **Alt+1**, siempre y cuando lo hayamos abierto desde el fichero *Template.pnproj*.

A lo largo de este libro usaremos tanto ejemplos de código fuente de programas como pseudocódigo para las explicaciones de las diferentes lecciones. El lector podrá encontrar esos ejemplos en <http://www.theNinjaBunny.com/libro/> o en el propio CD donde es distribuido (vaya como me flipo pensando hacer CD's de esto). Se recomienda que los ejemplos se instalen en */*carpeta de instalación de DevKitPro*/libro/*, para que compilen correctamente.

1.3. Especificaciones y limitaciones del hardware

A la hora de programar es importante conocer las limitaciones y características del hardware donde va a ser ejecutado, por ejemplo algo tan simple como saber cuales son las dimensiones de la pantalla donde se muestra la información o cuales son los dispositivos de entrada (botones, pantalla táctil, . . .). Para no aburrir excesivamente al lector, ahorraré la mayoría de esos detalles pro el momento y explicaré cada uno según sea necesario. Por el momento aclarar que la Nintendo DS refresca la pantalla a una velocidad de 60Hz, 60 veces por segundo, aunque si realizas demasiados cálculos y operaciones esa frecuencia puede disminuir y ralentizar la aplicación/juego. Tienes dos pantallas, la superior y la inferior, que es táctil. Dispone de 6 botones : A,B,X,Y como cruceta en la parte derecha y L,R como disparadores en la parte superior. Cuenta con un D-Pad que no es más que una cruceta para dirigirse a las 8 direcciones principales con el aliciente de que no se pueden pulsar simultáneamente izquierda-derecha o arriba-abajo, por lo que no nos encontraremos incómodas situaciones de que el usuario pulse izquierda y derecha al mismo tiempo y decidir para que lado ha de moverse nuestro personaje. Para terminar esta pequeña introducción, diré que tiene dos motores gráficos, uno de 2D bastante más limitado pero que será el que usemos; y otro para manejar las 3D bastante más flexible en cuanto a limitaciones pero que por el contrario nos obliga a trabajar a un refresco menor de pantalla (30Hz).

1.4. Hello Mario!

Porque una tradición es una tradición y ha de respetarse, en informática es común que el primer ejercicio a la hora de aprender/enseñar un nuevo lenguaje de programación sea imprimir en pantalla la frase "Hello World", así que démosnos a la tarea encomendada.

Copiamos la plantilla de PALib tal y como indicábamos unos párrafos antes, la podemos renombrar a *HelloMario* (el lector podrá encontrar este ejercicio en la carpeta *Cap01/cap01ej01/*). Luego abrimos el fichero *Template.pnproj* y seleccionamos el fichero *main.c* :

Cap01/cap01ej01/source/main.c

```

1 // Inclusiones
2 #include <PA9.h> // Incluyela libreria PALib
3
4 // funcion principal (esta es la llamada por el juego)
5 int main(int argc , char [] argv)
6 {
7     PA_Init() ; //Inici la PALib
8     //Inicia el VBL (sincronia con el reloj de la DS)
9     PA_InitVBL() ;
10
11     //Iniciamos el texto en el background 0 de la pantalla
12     //de arriba
13     PA_InitText(1,0) ;
14
15     //Escribimos "Hello Mario!" en la pantalla de arriba
16     //en la posicion 0,0
17     PA_OutputSimpleText(1,0,0,"Hello Mario!") ;
18
19     //Bucle principal de la aplicacion
20     while(1)
21     {
22         PA_WaitForVBL() ; //Esperamos un ciclo del
23         //procesador
24     }
25     return 0 ;
26 } //Fin de la funcion p r i n c i p a l

```

Sin duda el ejemplo más sencillo para empezar, pero analicemos línea a línea el código del programa, que además viene comentado.

```
#include <PA9.h> // Incluyela libreria PALib
```

Incluye la librería PALib, que a su vez carga las librerías libnds, las necesitamos si queremos que la cosa marche. Disponen de métodos y funciones que nos harán la vida bastante más fácil.

```
{
```

Esta línea es obligatoria en cualquier programa en C que realicemos, se encarga de iniciar la ejecución del programa. Es la primera función que es llamada, a partir de ahí nosotros podemos derivar el trabajo a otras funciones.

```
//Inicia el VBL (sincronia con el reloj de la DS)
```

Inicia las librerías PALib, es obligatoria si queremos usar esas librerías ya que hay que iniciar el hardware, variables y demás cosas en las que por el momento no nos interesa profundizar.

Iniciamos el reloj, la consola se refresca a 60Hz (Herzios) o ciclos por segundo, es decir, una acción en un bucle se ejecutará 60 veces en un segundo.

Iniciamos el fondo 0 de la pantalla superior para texto.

Escribimos texto en la pantalla superior (en el fondo 0 anteriormente iniciado) en la esquina superior izquierda.

```
{
```

Bucle infinito para mantener nuestra aplicación en marcha hasta que apagamos la consola.

```
    return 0 ;
```

Ejecutamos un "frame" o dejamos pasar un ciclo, de no ponerlo la consola intentaría ejecutar un bucle infinito en 0 segundos y se colgaría. Cada acción dentro del bucle ocurriría 60 veces por segundo, si escribimos dos veces seguidas `PA WaitForVBL()`; la acción se ejecutaría sólomente 30 veces por segundo.

A esta línea nunca deberíamos de llegar, significaría que salimos del bucle y que la aplicación se ha terminado.

Espero que hasta ahora la lectura haya sido amena y suficientemente clara y sencilla. En el próximo capítulo trabajaremos más a fondo en el tema de los textos, a mi entender lo más sencillo que se puede realizar programando.